Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

SDN: il futuro della rete
Stato dell'arte e casi reali

Relatore:
Chiar.mo Prof.
Fabio Panzieri

Presentata da:
Pietro Ridolfi

Sessione II
Anno Accademico 2014/2015

Ci sono 10 tipi di persone nel mondo:
coloro che capiscono il binario,
e coloro che non lo capiscono.

Indice

Capitolo 1

Introduzione

Il numero sempre crescente di utenti e applicazioni che si servono della rete Internet ha fatto sì che quest'ultima divenisse sempre più estesa e dunque complessa. Tuttavia, la struttura della rete è rimasta pressoché invariata.
Negli ultimi anni sono state portate avanti svariate ricerche e tentativi per cercare di migliorare suddetta struttura, ma con scarsi risultati data la complessità attuale di Internet e la difficoltà nell'apportarvi delle modifiche.
Qualche anno fa, però la Open Networking Founndation (ONF) ha avanzato una proposta che sembrerebbe costituire una vera svolta per la crescita e l'evoluzione dell'attuale rete di intercomunicazione.
Questa tesi vuole esporre le modalità attraverso le quali questa nuova architettura potrebbe potenzialmente cambiare il mondo informatico e quali sono i pro e i contro di questo approccio.
Già in molti hanno iniziato a sperimentare questo innovativo modello intuendone le potenzialità per portare Internet in una nuova era.

Più precisamente la tesi si prefigge l'obbiettivo di analizzare nel dettaglio questo emergente paradigma di rete, Software Defined Network, evidenziandone i punti di forza e mettendone quindi in luce i conseguenti vantaggi, le potenzialità, le limitazioni, l'attuabilità e i benefici, nonché eventuali punti di debolezza.

La sempre maggiore diffusione dei dispositivi mobili e l'utilizzo sempre più ampio dei servizi di cloud comporta una generale rivisitazione della tradizionale architettura di rete, la quale non ha subito sostanziali modifiche, ed è indi per cui rimasta invariata negli ultimi decenni.
Le architetture classiche sono tendenzialmente gerarchiche e statiche, il che porta a dover affrontare problematiche non indifferenti, in un contesto in cui la dinamicità e la flessibilità sono la condizione necessaria e imprescindibile al progresso.
Analizziamo alcuni motivi [1] che inducono a pensare di adottare un nuovo approccio:

Problemi di traffico di dati

La modalità di circolazione del traffico dati è opportunamente cambiata. Nei data center infatti, tali modelli di trasporto sono stati rivoluzionati, e contrariamente a quanto succedeva con il classico sistema client-server, nel quale appunto le comunicazioni avvenivano tra un client e un server, le applicazioni odierne accedono a differenti database e server, creando molto traffico dati intermedio tra diverse macchine, prima che il risultato della richiesta sia inoltrato al dispositivo dell'utente.
Allo stesso tempo anche gli utenti hanno cambiato il modo di usufruire dei servizi di rete, avendo la possibilità di accedere a dati e applicazioni da qualsiasi dispositivo, connesso da qualunque luogo in ogni momento.
Per questo molti amministratori di data center hanno pensato di usare servizi di cloud, causando in questo modo un aumento di traffico attraverso la rete WAN (wide area network).

Traffico mobile

Nell'attuale era di Internet, l'accessibilità ai servizi della rete è decisamente più immediata rispetto agli albori: con l'avvento di apparecchi quali

tablet e smartphones, gli utenti hanno la possibilità di utilizzare tali dispositivi personali per accedere alle reti aziendali.
Per questo motivo ai tecnici informatici spetta l'arduo compito di permettere a questi dispositivi l'accesso alle risorse aziendali garantendo però l'integrità e la riservatezza dei dati sensibili, oltre che le proprietà intellettuali.
Bisogna ad ogni modo tenere a mente che queste operazioni devono garantire prestazioni ottimali rispettando i requisiti di conformità.

Cloud Computing

La messa a punto del paradigma di rete Cloud computing, ha comportato dei cambiamenti notevoli nel panorama informatico.
Le imprese, intuendone i vantaggi, lo hanno accolto con entusiasmo e favore, sfruttandone proficuamente i servizi. Di conseguenza oggi assistiamo ad una esponenziale crescita di quest'ultimi.
Tuttavia, insieme alle possibilità economiche e organizzative, anche l'aspettativa delle aziende è cresciuta: la pretesa è di accedere agevolmente alle applicazioni, alle infrastrutture e ad altre risorse informatiche on-demand.
A complicare ulteriormente la situazione, va detto che la pianificazione dei servizi cloud deve necessariamente essere compiuta in un ambiente il più possibile sicuro e conforme ai requisiti aziendali, che possono comunque cambiare drasticamente in seguito a riorganizzazioni, consolidamenti o fusioni.
Provvedere a fornire servizi su misura e su richiesta, sia in un cloud pubblico che privato, richiede dunque la dovuta elasticità, in modo da poter scalare agilmente, utilizzando semplici tool, capacità di calcolo, memoria e risorse di rete.

File di grandi dimensioni

Per ciò che concerne la gestione dei file di grandi dimensioni o di grandi dataset, largamente diffusi al giorno d'oggi, è richiesta l'esecuzione contemporanea di molti algoritmi paralleli su migliaia di server distinti, i quali devono poter comunicare tra loro. L'aumento di suddetti dataset comporta di con-

seguenza la costante crescita della domanda per migliorare le prestazioni, ma soprattutto la capienza della rete (bandwidth).
Gli operatori dei data center devono quindi adattare la rete a dimensioni inimmaginabili, mantenendo però le connessioni tra tutti i dispositivi e garantendone il funzionamento ottimale.

L'architettura Internet oggi

L'attuale architettura non è pensata per soddisfare le odierne esigenze degli utenti.
Le aziende di IT riscontrano sempre maggiori problemi, nei quali una non indifferente fetta di budget viene investita per farvi fronte, causando perdite di profitti. Si trovano inoltre a dover sostenere costi elevati di gestione, data la struttura antiquata e inadatta di Internet, per soddisfare le domande di mobilità e banda sempre crescenti.

Protocolli ad-hoc

La tecnologia del network finora si è basata su un gran numero di protocolli di rete, progettati per connettere host in modo affidabile su distanze, velocità di connessione e topologie variabili. Per soddisfare le esigenze commerciali e tecniche degli ultimi decenni l'industria ha sviluppato protocolli network per permettere performance migliori, alta affidabilità, connettività più ampia e una maggiore sicurezza.
Tali protocolli solitamente sono sviluppati individualmente, ed ognuno risolve un problema specifico, senza pertanto usufruire dei benefici di un piano di astrazione ulteriore.
Questo modus operandi ha reso la rete notevolmente complessa, costituendone uno dei maggiori limiti. Ad esempio per aggiungere o rimuovere un qualsiasi dispositivo dalla rete, è necessario modificare diversi switch, router, firewall, portali di autenticazione web, aggiornare le ACL e altri meccanismi basati sui protocolli citati, usando tool di basso livello.

Eterogeneità dei nodi

Bisogna poi anche tenere conto della posizione fisica degli switch, del loro modello e della versione del software istallata. Questa configurazione porta la rete ad essere piuttosto statica, considerando anche che la principale preoccupazione degli addetti ai lavori è quella di non far mai cadere il servizio.

Virtualizzazione

La natura rigida dell'architettura del network è in netto contrasto con la natura dinamica degli ambienti server, dove la virtualizzazione ha aumentato in modo consistente il numero di host che richiedono connettività di rete e fondamentalmente ha alterato le assunzioni sulla locazione fisica di quest'ultimi. Prima della virtualizzazione le applicazioni risiedevano in un singolo server e scambiavano traffico con i client selezionati, contrariamente alle attuali applicazioni, le quali sono distribuite su diverse macchine virtuali che scambiano flussi di dati tra di loro.
Il passaggio alle macchine virtuali ha ottimizzato e bilanciato i carichi di lavoro dei server, permettendo anche di modificare la posizione fisica delle macchine. Questo approccio mette a dura prova alcuni aspetti della tradizionale architettura di Internet, dall'indirizzamento dei pacchetti al modello di routing.
Inoltre per adottare queste tecnologie di virtualizzazione molte imprese si servono di un IP converged network, ossia una rete per trasportare diversi tipi di traffico dati, quali voce, dati e video in un'unica rete senza doverne costruire diverse per ogni tipo di traffico.

Diversi livelli di QoS

Benché l'attuale rete può fornire differenziati livelli di Quality of Service (QoS) per varie applicazioni, la configurazione delle risorse necessarie è molto manuale e dunque più intricata. Gli informatici devono infatti configurare ogni dispositivo delle diverse marche in modo separato e sistemare parametri,

come ampiezza di banda e QoS, per ogni sessione e per ogni applicazione. Questo per via della già citata staticità della rete, che non possiede i mezzi per potersi adeguatamente adattare in modo dinamico al cambiamento del traffico, alle differenti applicazioni ed alle richieste degli utenti.

Riconfigurazione della rete

Per implementare delle direttive su vasta scala, gli informatici potrebbero dover configurare migliaia di dispositivi.
Ad esempio ogni volta che viene aggiunta una nuova macchina virtuale all'impianto, potrebbero servire ore, in alcuni casi anche giorni, per riconfigurare le ACL nell'intera rete. La complessità delle reti attuali rende difficile per i tecnici applicare un vasto set di politiche di accesso, sicurezza, QoS e altre direttive per avvicinarsi ai mobile user.
Questo comporta dei buchi di vulnerabilità e la non inerenza agli standard.

Problemi di scalabilità

Alla rapida diffusione dei data center è corrisposta una speculare crescita della rete. Tuttavia quest'ultima è divenuta molto più complessa con l'aggiunta di un numero spropositato di altri dispositivi, che bisogna opportunamente configurare e gestire.
Grandi aziende come Google, Yahoo! e Facebook, devono affrontare problemi di scalabilità ancora maggiori. Questi colossi commerciali impiegano algoritmi paralleli su larga scala associati a dataset sparsi sull'intera rete.
Con la sempre più consistente portata delle applicazioni end-user, il numero degli elementi di rete sale vertiginosamente e lo scambio di dati tra i vari nodi dei data-set può raggiungere petabytes. Queste grandi aziende hanno bisogno delle cosiddette hyperscale networks che sono in grado di fornire alte prestazioni e connessioni a basso prezzo tra i centinaia di migliaia di server fisici. Non è in alcun modo pensabile compiere un'espansione di tale portata tramite una configurazione manuale. Per far fronte alla concorrenza, questi

giganti informatici devono fornire servizi differenziati garantendo sempre il massimo della qualità ai clienti.
La molteplicità di fruitori con diversi bisogni in fatto di applicazioni e performance richieste, complica ulteriormente il lavoro.
Nonostante una potenziale soluzione sia stata pensata (indirizzare il traffico dei clienti in flussi che permettano controlli personalizzati in base alle richieste), è difficilmente attuabile, specialmente trattandosi ipoteticamente di operare in scala così grande, poiché data l'incompatibilità con l'attuale architettura di rete, richiederebbe hardware specifico, implicando un incremento dei costi ed un aumento dei tempi per introdurre nuovi servizi.

Guerra degli standard

Le aziende cercano di implementare nuovi servizi in rapida risposta alle mutevoli esigenze degli utenti, ma la loro capacità nel rispondere prontamente a questi bisogni è condizionata dalle limitazioni imposte dai produttori di infrastrutture, che tendono ad impiegare anni per adeguare gli strumenti all'innovazione.
La mancanza di standard e interfacce open-source, limita la capacità degli operatori di rete di modellare la stessa ad hoc per i propri fini. Il divario tra i requisiti del mercato in rapida crescita e le inadeguate competenze della rete ha portato ad un punto di stallo.

In risposta alle problematiche finora esposte, la ONF ha creato il Softaware-Defined Network(SDN) e lo sviluppo di standard associati ad esso.
SDN rappresenta una novità rivoluzionaria nell'attuale panorama informatico.
Questa innovativa architettura di rete è dinamica, di facile gestione, economicamente vantaggiosa e adattabile; questo la rende ideale per gestire le applicazione di oggi, che sappiamo essere di natura dinamica e richiedere molta banda.

Considerate le già discusse limitazioni dell'attuale architettura di rete e le potenzialità di SDN molte università hanno iniziato a condurre esperimenti su questo innovativo paradigma. Anche diverse grandi aziende come Google sembrano avere abbracciato l'approccio di SDN, intuendo notevoli margini di miglioramento sia in termini di facilità di gestione che di risparmio economico.
Tuttavia questa neonata tecnologia è ancora in una fase di sviluppo e test, benché ci sia un ottimismo diffuso in tutto l'ambiente, che considera questo modello come la vera svolta per la gestione di Internet. Bisognerà attendere ancora qualche anno di sperimentazioni per vedere totalmente sfruttate le potenzialità di SDN.

Lo studio proposto in [6] ritiene opportuno fornire un breve ma esaustivo elenco di possibili casi d'uso di SDN:

- I multi-tenant data center hanno bisogno di fornire diversi servizi per diverse richieste di vari clienti. SDN data la sua flessibilità potrebbe agevolmente provvedere a questa differenziazione dei servizi.

- Multiplayer Online Role Playing Games hanno bisogno di risorse di rete che sono notevolmente soggette a variazione. Anche in questo caso SDN aiuterebbe nella gestione di questi cambi repentini di richieste da parte dei client.

- Internet of Things sembra essere un grande trascinatore per lo sviluppo futuro di Internet, avrà bisogno di funzionalità specifiche come ad esempio security access, delay bounding access per qualche sensore ed altre funzionalità che potrebbero venire fuori nello sviluppo futuro di questa tecnologia. SDN potrebbe essere di grande aiuto nella realizzazione di queste funzionalità in modo facile e flessibile.

- I Content Distribution Networks (CDN) data la loro estensione hanno bisogno di avere una gestione più semplificata e questo può facilmente essere raggiunto con l'utilizzo dell'architettura SDN.

- L'aumento di tutte le middlebox in Internet ha reso difficile lo sviluppo di altri protocolli di trasporto, la creazione di molte funzionalità ad-hoc. SDN può dare un sistema programmabile open-source e con API pubbliche in modo da avere un modello più generale da poter usare in ogni situazione senza dover più creare delle reti su misura per un determinato servizio.

- La flessibilità nelle attuali reti è ottenuta con approcci ad-hoc per consentire replicazione. SDN può aiutare a unificare una serie di meccanismi sotto un unico control plane.

- SDN può portare un aumento della sicurezza. Usando linguaggi di programmazione più sicuri, tecniche di analisi dei flussi, verifica software, ecc...

- la gestione di grandi reti private che devono avere servizi di rete su misura che richiedono un ingente esborso di denaro. SDN offre un modo di costruire una rete più flessibile che può adattarsi alle esigenze delle aziende in modo più semplice e con un costo molto minore.

Capitolo 2

Software-Defined Networking

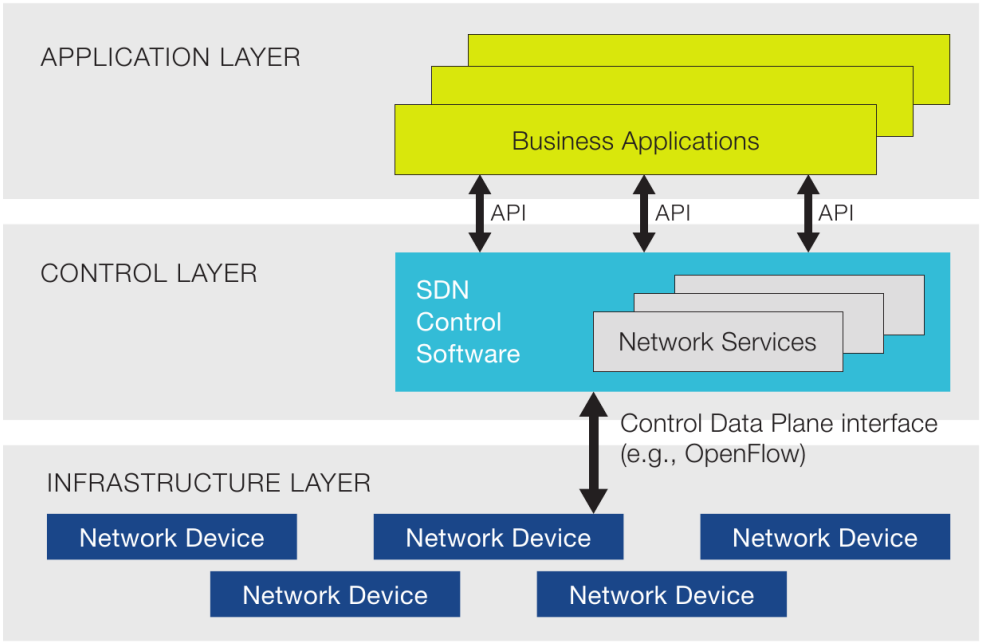

Figura 2.1: Architettura SDN [1]

L'architettura SDN come proposta in [1], si caratterizza principalmente nella divisione tra data plane e control plane, ed è composta da tre strati distinti:

- Application Plane: consiste nelle applicazioni utenti e comunicano con il piano sottostante tramite delle API.

- Control Plane: tutta l'intelligenza di rete è centralizzata in questo livello, il controller ha una vista globale sulla rete e decide le politiche di forwarding dei pacchetti del *data plane*, occupandosi di riempire le tabelle.

- Data Plane: consiste negli elementi della rete che consentono l'inoltro fisico dei pacchetti(switch). Più precisamente il *data plane* si preoccupa di fare il forwarding dei pacchetti usando le tabelle di forwarding fornite dal *control plane*. Questo processo semplifica enormemente la logica del *data plane*, riducendo la complessità e il costo degli switch.

Uno dei principi chiave di SDN è che le sue applicazioni sono network aware, cioè hanno la possibilità di conoscere lo stato della rete a differenza delle reti tradizionali che tipicamente richiedono molti controlli umani come le politiche di controllo e negoziazione nel caso in cui non ci siano sufficienti risorse. Inoltre le reti odierne come Internet o i servizi di streaming non offrono un modo dinamico per esprimere una vasta gamma di requisiti lato utente come il throughput, delay, variazioni di delay o la disponibilità. Gli header dei pacchetti possono contenere richieste con priorità ma generalmente vengono ignorati dai network provider per questioni di sicurezza. SDN offre all'utente la possibilità di specificare completamente i suoi bisogni in un contesto affidabile.

Centralizzazione del piano di controllo

Negli anni 60 fu creato ARPANET, pensato come piano d'emergenza per lo scambio di messaggi in caso di attacco. ARPANET presentava un'architettura distribuita, che permetteva ai router di ridirezionare i pacchetti qualora uno di essi non fosse operativo. Sia il piano dati che il piano di controllo erano distribuiti: ogni router si occupava di trovare la route di un pacchetto,

scambiando informazioni con gli altri router.
Internet ha ereditato lo stesso tipo di paradigma di distribuzione. La centralizzazione era infatti considerata un male fino a pochi anni fa, mentre adesso se ne sono rivalutati i vantaggi: la centralizzazione del controllo permette di rilevare lo stato della rete e di regolare le politiche di forwarding dinamicamente in base ai cambiamenti di stato, molto più velocemente rispetto ai protocolli distribuiti.
Un altro grande vantaggio della centralizzazione del controllo è che se lo stato o la policy di gestione cambiano, la propagazione delle nuove informazioni è molto più rapida di un sistema distribuito. Per gestire eventuali failures di controller principali possono essere predisposti controller in stand by. Si noti che il piano dati è ancora completamente distribuito, mentre quello di controllo può non essere fisicamente centralizzato: per questioni di performance, scalabilità e di fiducia la logica centralizzata del controller SDN può essere distribuita su più controller fisici che cooperano al controllo della rete e delle applicazioni. Dunque nell'architettura basata su SDN il controller mantiene una vista globale della rete, che appare alle applicazioni come un singolo switch logico.

Programmabilità del piano di controllo

La centralizzazione del piano di controllo di SDN, come illustrato dettagliatamente in [7], permette una maggiore facilità di inserimento di modifiche da parte degli amministratori di reti, tramite la modificazione del programma di controllo che è scritto in un linguaggio di alto livello, così da garantire maggiore sicurezza e facilità di gestione. È inoltre possibile ottimizzare le risorse di rete in modo più immediato, attraverso programmi automatizzati che i network manager stessi possono scrivere, poiché i programmi non dipendono da software proprietario. Con le API corrette si potrebbe implementare una varietà di policy e cambiarle dinamicamente a seconda del variare dello stato e delle esigenze. Ciò rende la rete agile e permette di gestire anche cambiamenti repentini dello stato della rete o delle richieste

delle applicazioni. Questo costituisce l'aspetto cruciale di SDN. Un piano di controllo programmabile permette di dividere la rete in più reti virtuali con policy diverse anche su hardware condiviso, mentre con un piano di controllo distribuito la dinamicità di queste ultime sarebbe troppo lenta e complessa.

API standard

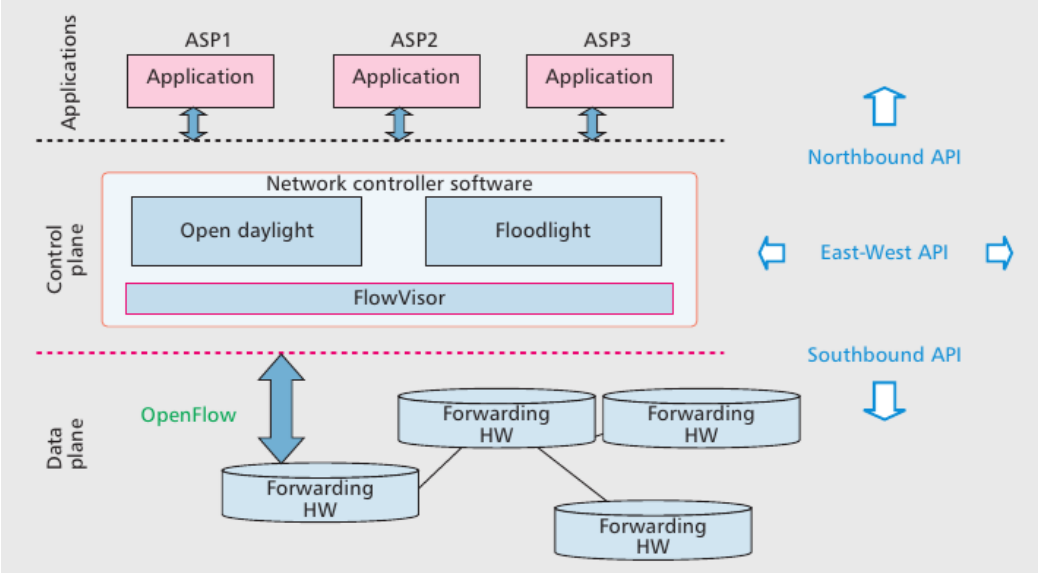

Figura 2.2: API SDN [7]

Le API standard di SDN descitte in [7] si dividono in due macro-gruppi: northbound e southbound; le prime permettono la comunicazione delle applicazioni con il controller, mentre le seconde interagiscono con l'hardware sottostante.

In ambito scientifico si dispone già di diversi controller, quali ad esempio Floodlight, OpenDaylight o FlowVisor, il quale agisce in modo trasparente come proxy tra hardware e gli altri controller.

Per quanto riguarda le API southbound, sebbene non sia indispensabile per l'architettura SDN, OpenFlow è ormai diventato lo standard de-facto, sviluppato dalla Open Networking Foundation. Mentre per quanto riguarda le API northbound non vi è ancora uno standard, ogni controllore può avere diverse interfacce programmabili: questo rende lo sviluppo di applicazioni

per SDN più difficoltoso.
L'esistenza di diverse tipologie di controller introduce l'esigenza di sviluppare API East-West per la loro interazione.
Bisogna che le API siano rigorosamente open, in modo da evitare che il software da sviluppare sia legato alle specifiche di un singolo produttore.

2.1 Architettura SDN

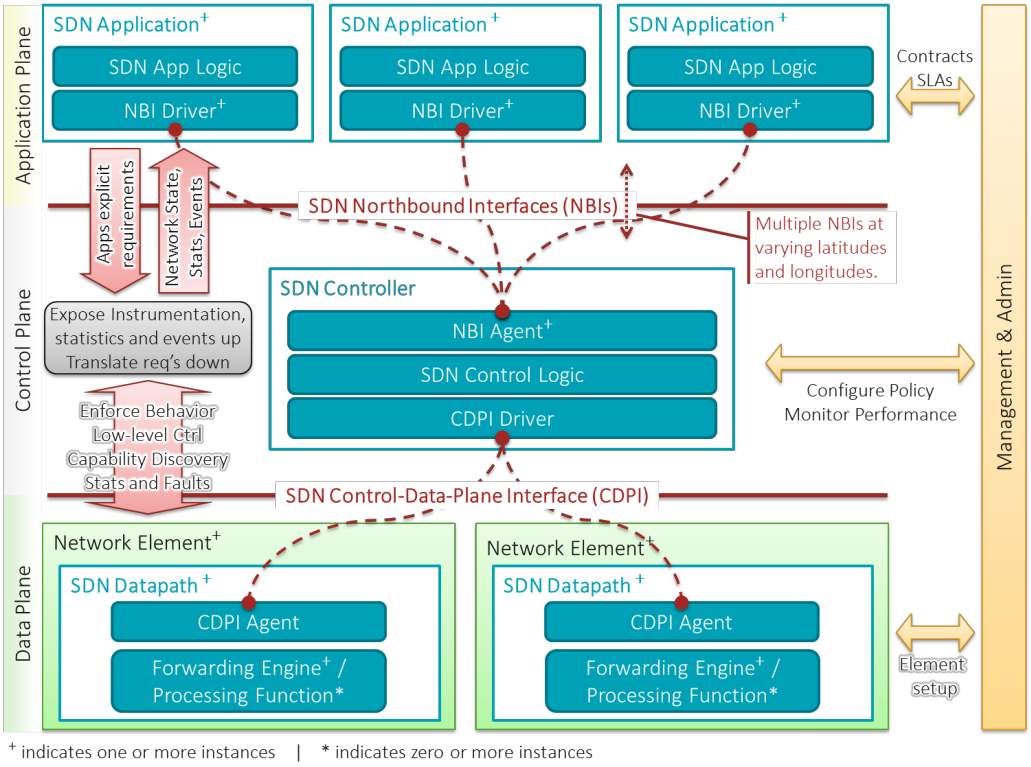

Figura 2.3: Architettura SDN in dettaglio [2]

Come precedentemente accennato, nell'architettura SDN [2] si possono identificare tre livelli o piani: quello dati (data plane), quello di controllo (control plane) e quello applicazioni.
Il livello più basso è quello dei dati, che comprende SDN Datapaths nel quale agisce il Control-Data-Plane Interface Agent (CDPI); il livello più alto invece è quello in cui risiedono le applicazioni SDN, che si interfacciano mediante

i Northbound Interface Drivers (NBI); infine, in quello centrale, il controller SDN si preoccupa di tradurre le richieste dell'SDN Datapath alle applicazioni SDN.

Applicazioni SDN

Per quanto riguarda le applicazioni SDN, si tratta di programmi che comunicano dinamicamente, esplicitamente e direttamente i requisiti di rete e il suo stato al controller SDN attraverso le NBI. Un'applicazione SDN consiste in un SDN application logic e uno o più NBI driver .

Controller SDN

Il controller SDN è un entità logica centralizzata che si occupa di tradurre i requisiti del livello applicativo al datapath SDN e fornisce alla rete un livello di astrazione delle applicazioni, come statistiche ed eventi. Più precisamente, un controller SDN consiste in uno o più agenti NBI, un SDN control logic e i driver CDPI.
La sua logica centralizzata non dipende dai dettagli di implementazione, quali la gestioni di controller multipli, l'ereditarietà della connessione dei controller, le interfacce di comunicazione tra controller, la virtualizzazione o la divisione delle risorse di rete.

SDN Datapath

Il Software Defined Network Datapath è un device logico di rete che si occupa del corretto forwarding e data processing dei pacchetti. La sua rappresentazione logica può sia comprendere tutte le risorse fisiche sottostanti, che solo una parte di esse.
Un SDN Datapath consta di: un agent CDPI, uno o più set di gestori del traffico e zero o più funzioni di traffic processing. Queste funzioni e gestori del traffico, permettono l'inoltro di pacchetti tra le interfacce esterne del datapath, il flusso del traffico interno e le funzioni di terminazione. Uno o più

datapath SDN possono essere contenuti in una singola entità fisica di rete. La sua logica non dipende da dettagli di implementazione, come il mapping logico-fisico, la gestione delle risorse fisiche condivise, la virtualizzazione, la suddivisione del datapath SDN, l'interoperabilità con reti non SDN o le funzioni di data processing.

Il controller SDN ha un controllo totale dei datapath SDN e non interferisce con gli altri elementi del piano di controllo, il che semplifica lo scheduling e la locazione delle risorse; questa caratteristica consente alle reti di funzionare con complesse e precise policy con un ottimo utilizzo delle risorse, garantendo la QoS.

SDN Control to Data-Plane Interface (CDPI)

Il Software Defined Network CDPI è un'interfaccia, definita tra un controller SDN e un datapath SDN, che fornisce un controllo programmatico di tutte le operazioni di forwarding, la possibilità di conoscere le capabilities e rende disponibili report statistici e notifiche di eventi. Uno dei punti chiave di SDN consiste nell'aspettativa che il CDPI sia implementato in modo open, affinché esso risulti interoperabile e non legata ai produttori.

SDN Northbound Interfaces (NBI)

Le NBI SDN sono interfacce tra le applicazioni SDN e i controller SDN e solitamente forniscono un livello di astrazione della rete, permettendo lo scambio di richieste. Anche in questo ambito è cruciale che le interfacce siano di natura open, interoperabili e non legate ai produttori.

Interface Drivers e Agenti

Ogni interfaccia è implementata da una coppia di agenti, uno per gestire le richieste hardware e uno per quelle applicative.

2.2 OpenFlow

OpenFlow [1, 3] è il primo standard di comunicazione definito come interfaccia tra il piano di controllo e il piano di forwarding di una architettura SDN; fornisce accesso diretto e permette di manipolare il piano di forwarding dei device di rete come switch e router sia fisici che virtuali(basati su hypervisor). L'assenza di un'interfaccia open verso il forwarding plane ha portato a quello che sono i device di rete oggi, ovvero monolitici, chiusi e simili a main-frame. Nessun altro protocollo standard fa quello che fa openflow e un protocollo come openflow è necessario al fine di spostare il controllo di rete dagli switch a un'unità logica centralizzata.
OpenFlow può essere comparato all'istruction set della CPU come mostrato in figura il protocollo specifica primitive che possono essere usate da applicazioni esterne per gestire il forwarding plane dei device di rete, come fanno analogamente le instruction set della CPU ed un architettura di sistema.

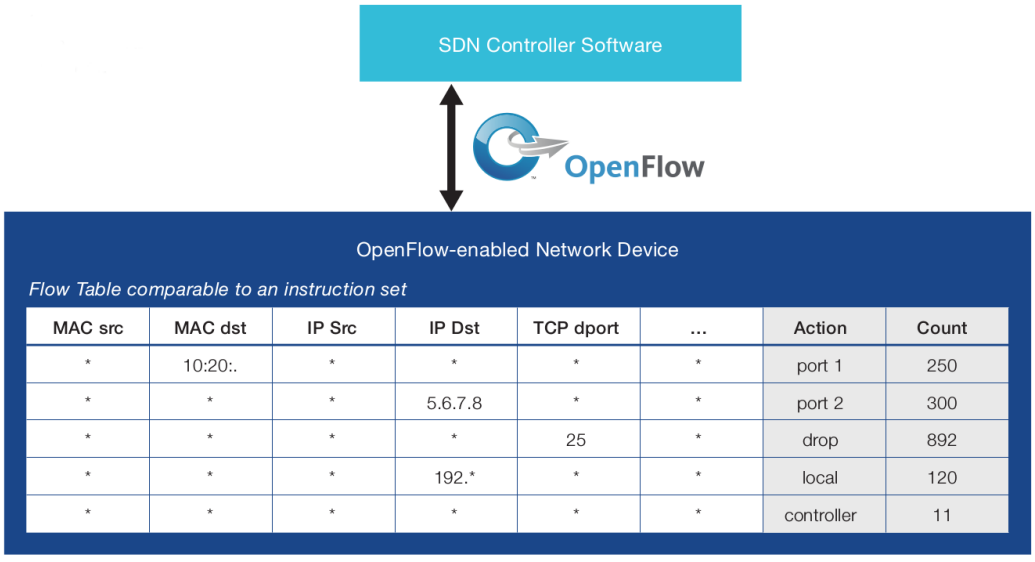

Figura 2.4: Flow table [1]

OpenFlow usa il concetto di flusso per identificare il traffico rete basandosi su predefinite regole di matching che possono essere statiche o dinamiche

nelle applicazioni SDN. Inoltre permette di definire come il traffico debba essere gestito basandosi su parametri come i pattern d'uso, le risorse delle applicazioni o quelle cloud. L'architettura SDN basata su OpenFlow fornisce un controllo estremamente granulare permettendo alla rete di rispondere in tempo reale ai cambiamenti a livello user, applicazione o sessione.
i meccanismi di routing di oggi basati su IP, non forniscono questo tipo di controllo dato che tutti i flussi tra due end-point devono seguire lo stesso path lungo la rete.
il protocollo OpenFlow è un punto chiave della rete SDN e al momento è l'unico protocollo SDN standard che permette la manipolazione diretta del forwarding plane. Inizialmente openflow era usato solo su reti basate su ethernet, gli switch openflow possono essere estesi ad una moltitudine di casi d'uso. Le SDN basate su openflow possono essere sviluppate su reti esistenti sia fisiche che virtuali. I device di rete possono supportare il forwarding gestito da openflow come con il forwarding tradizionale, il che rende tutto più facile per le imprese che vogliono adottare questa nuova tecnologia. Infatti openflow è largamente adottato da aziende di infrastrutture che l'hanno tipicamente implementato attraverso semplici aggiornamenti firmware o software.

Benefici di una rete SDN basata su OpenFlow

La facilità di implementazione di SDN non grava sui costi delle aziende che gestiscono reti; l'introduzione di openflow inoltre permette agli IT di incontrare le esigenze di molta banda delle applicazioni di oggi, di adattare la rete alla loro natura dinamica e riduce in modo consistente la complessità di gestione. I maggiori benefici derivanti da reti SDN basata su OpenFlow sono:

Centralizzazione del controllo in un ambiente eterogeneo

Un controller SDN può controllare indistintamente ogni device OpenFlow-enabled di produttori diversi, inclusi switch, router e switch virtuali. Gli

amministratori di rete non hanno bisogno di gestire gruppi di device in base al produttore ma possono gestire tutti gli elementi di rete usando tool basati su SDN che permettono la gestione simultanea dell'intera rete; in modo da poter configurare ed aggiornare in modo semplice e diretto ogni tipo di device.

Complessità ridotta mediante automatizzazione

Le reti SDN basate su OpenFlow permettono un uso flessibile della rete e, mediante framework di gestione dedicati, consentono di realizzare tool dedicati all'automatizzazione di numerosi task che ancora oggi vengono svolti manualmente. Questa automatizzazione riduce considerevolmente l'overhead operazionale e l'instabilità di rete causata da errori umani e si ben adatta al concetto emergente di IT-as-a-Service o del self-service provisioning.

Alto tasso di innovazione

L'adozione di SDN rende estremamente efficiente l'intera gestione di rete da parte degli amministratori, permettendo di modellarla in real-time venendo incontro a specifiche esigenze commerciali o alle richieste sollevate dagli utenti. Attraverso la virtualizzazione e l'astrazione di rete, i fornitori possono modellare su misura i servizi offerti, possono introdurre nuovi servizi e cambiare capabilities, il tutto in poche ore, mentre per le tradizionali reti occorrerebbero giorni.

Incremento dell'affidabilità e della sicurezza

SDN permette agli amministratori di definire configurazioni ad alto livello e policy adeguatamente tradotte in istruzioni OpenFlow. L'architettura SDN basata su OpenFlow elimina la necessità di configurare singolarmente ogni device ogniqualvolta che viene aggiunto/tolto un end-point, un servizio o un'applicazione, o se viene cambiata qualche policy di gestione; si riduce così il rischio di errori di rete dovuti a inconsistenze nelle policy o nelle configurazioni. Dato che i controller SDN forniscono una visibilità completa,

permettono di interagire su qualunque aspetto del controllo degli accessi, del traffico, del QoS, della sicurezza o delle gestione delle policy.

Controllo di rete granulare

Il controllo del flusso basato su OpenFlow fornisce agli amministratori la possibilità di applicare policy a un livello di granularità molto alto, potendo definire regole per ogni sessione, per utenti diversi, per molteplici device o per differenti applicazioni, tutto in modo automatico e ad un alto livello di astrazione. Questo controllo permette inoltre la multi-tenancy in ambienti cloud, mantenendo comunque il traffico isolato, la sicurezza e l'elasticità nella gestione delle risorse in un'architettura condivisa per più utenti.

User expericence migliore

Centralizzando il controllo di rete e rendendo lo stato disponibile ad alto livello alle applicazioni, un'architettura SDN si adatta dinamicamente alle esigenze degli utenti. Per esempio un fornitore potrebbe voler introdurre un servizio di streaming video che permetta agli utenti, che sottoscrivono un abbonamento, una migliore risoluzione in modo automatico e trasparente. Oggi gli utenti sono costretti a impostare manualmente le impostazioni di risoluzione introducendo ritardi e interruzioni che gravano sulla user-experience. Con una rete SDN basata su OpenFlow, l'applicazione video rileverebbe automaticamente la velocità di bandwith adatta alla rete in tempo reale, aggiustando le impostazioni adeguatamente.

Capitolo 3

Casi d'uso

In questo capitolo verranno presi in esame alcuni casi d'uso concernenti le reti SDN.

3.1 AgNOS: A Framework for Autonomous Control of Software-Defined Networks

In questa sezione parleremo dell'esperimento condotto dalla Institute of Computing Federal University of Amazonas Manaus, AM - Brasile, che ha realizzato un framework, AgNOS [5], per il controllo autonomo delle reti SDN, cercando di capirne l'efficacia nel limitare uno scenario di un attacco Distribuited Denial of Service.

Introduzione

Le reti si dovrebbero auto regolamentare in caso di malfunzionamenti. Si ipotizza così l'introduzione di un terzo piano chiamato knowledge plane [10] che poggia su una AI (Artificial Intelligence) e su tecniche di sistemi cognitivi.
Questo tipo di rete dovrebbe essere auto-cosciente del proprio stato, capace di riassemblarsi in caso di richieste di cambiamento, capace di rilevare errori

e riparare i failures. Il modo più consono per costruire un layer cognitivo è quello di utilizzare agenti intelligenti [11] (intelligent agents). Aldilà dei vantaggi derivati di reattività (reactivity) e pro-activity, gli agenti intelligenti hanno anche abilità sociali [12], come cooperazione e negoziazione al fine di far collaborare domini di rete. Gli approcci correnti di gestione autonoma mancano di meccanismi per gestire la conoscenza riguardo la rete se non utilizzando complesse informazioni a basso livello; ad oggi gli AS (Autonomous Systems) non sono adattabili a reti a larga scala come Internet.
Per questo SDN rappresenterebbe una soluzione in quanto permette alle applicazioni di essere sviluppate in modo centralizzato e scritte con astrazioni ad alto livello, invece di usare parametri a basso livello come fa l'attuale architettura Internet. Le applicazioni di gestione mantengono delle map nome/indirizzo e monitorano i cambiamenti di traffico della rete. Queste applicazioni permettono una visione centralizzata dei cambiamenti di rete.
L'astrazione fornita da SDN sembra essere la maniera più giusta per riuscire a creare un'architettura agent-based per controllare e gestire al meglio gran parte della futura rete Internet. Molte delle limitazioni delle reti autonome di oggi possono essere risolte utilizzando funzioni presenti nel network controller. La figura 1 mostra la relazione tra le debolezze del framework ad agenti e le funzioni di SDN.

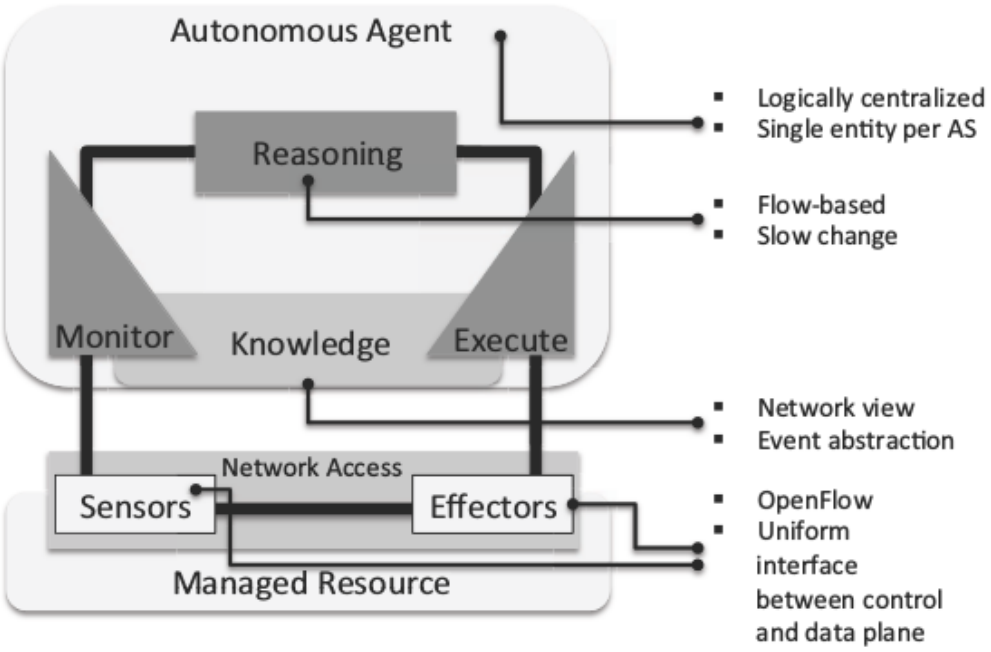

Figura 3.1: Interazione Autonomous Agent con rete SDN [5]

La rappresentazione della conoscenza dentro un framework ad agenti è totalmente dipendente dalla vista della rete che ha il network controller. Questa vista permette agli agenti di modellare diverse entità di rete come protocolli, pacchetti, percorsi di instradamento, access control lists, utenti e servizi.

In questo modo è possibile creare una rappresentazione concisa senza aver bisogno di gestire entità a basso livello come indirizzi IP, indirizzi MAC e header di pacchetti dei protocolli. Avere una conoscenza(knowledge) ad un alto livello di astrazione riduce l'onere di manipolare un numero esponenziale di fatti nella Knowledge base (insieme delle azioni che l'agente può fare in base ai dati che ha a disposizione) dell'agente.

Anziché ragionare su una logica pacchetto per pacchetto, gli agenti agiscono sul flusso di pacchetti. Questo significa che un agente gestisce l'inizializzazione del flusso (il primo pacchetto in un flusso) e pochi altri pacchetti legati ad eventi rilevanti.

In SDN gli agenti non hanno accesso diretto alle risorse di rete, dato che non hanno bisogno di implementare interfacce diverse per ogni dispositivo di rete con cui hanno a che fare. Viene utilizzata solo una interfaccia, ossia il protocollo OpenFlow, che permette l'accesso alle entità di rete. Quindi eventi della rete e azioni sulla rete sono gestite tramite OpenFlow. Inoltre non c'è bisogno di implementare codice di esecuzione agente per ogni router o host nella rete. Questo riduce i problemi di esecuzione su ambienti diversi o attacchi introdotti da questi frameworks.
Gli agenti sono sviluppati direttamente sui network controller. Invece avere agenti distribuiti su ogni nodo della rete, sono logicamente centralizzati nel cervello della rete. Per una rete di larga scala come Internet, ogni AS ha il proprio network controller che lo supporta. In questo modo si ridurrebbero il numero di agenti necessari da milioni a migliaia, cioè il numero di AS in Internet. Così facendo è possibile ridurre la complessità del protocollo di interazione tra AS e vi sarebbe la riduzione del traffico necessario alla gestione degli stessi.
Il control plane offre efficienti mezzi per la cooperazione inter-dominio, coordinazione e negoziazione. L'interfaccia centralizzata sviluppata da SDN permette un'astrazione che riduce la complessità di rappresentazione e la capacità di apprendere della knowledge di rete da parte degli agenti. Si potrebbe così arrivare ad avere reti SDN autonomamente controllate su vasta scala.

AgNOS: Agent-based Network Control

Sviluppare software ad hoc per mitigare DDoS o per gestire il bilanciamento del traffico è estremamente complesso, soprattutto per reti poco flessibili e dinamiche. Però un approccio basato su agenti è utile e funzionale dato che risolve molte problematiche.
Per far interagire correttamente gli agenti con la rete SDN vi è la necessità di un linguaggio ad alto livello insieme a un motore inferenziale, ovvero un linguaggio formale al fine di ottenere deduzioni corrette basate su sentenze iniziali. Inoltre l'agente si occuperà delle azioni classiche dell'intelligenza

artificiale come l'apprendimento, l'elaborazione dei dati percepiti e l'azione sull'ambiente, in concomitanza con l'interazione con altre reti SDN. La componente SDN di un agente è composta da due livelli di programmazione: - Uno per la logica deduzionale riguardo alle specifiche dichiarative - Uno per la programmazione imperativa di azioni
Nella prossima sezione verrà descritto un framework per creare reti autonome. Il framework prende il nome di Agent for Network OS (AgNOS).

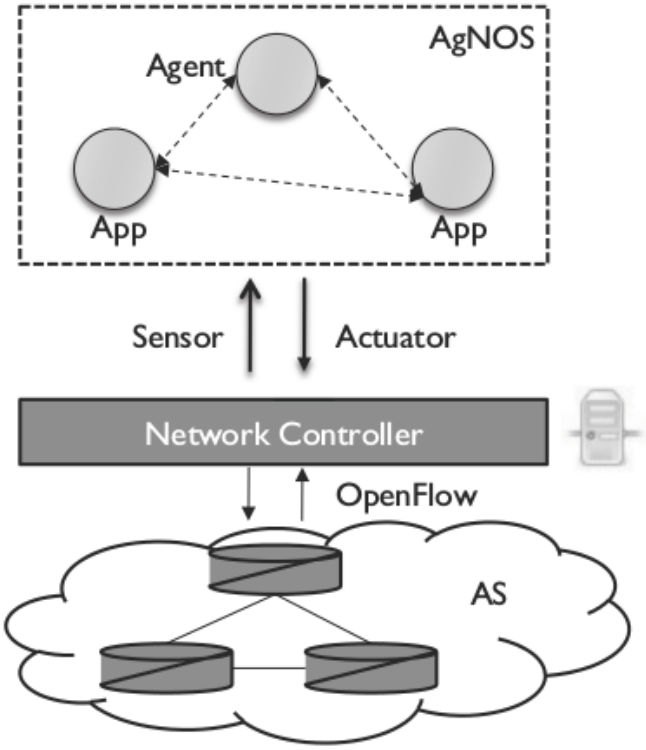

Figura 3.2: AgNOS framework [5]

AgNOS: elementi base

Un'azione di un agente AgNOS può essere di due tipi: succeed o fail; questa ha effetti sull'ambiente mediante eventi generati o messaggi che vengono mandati al soggetto dell'azione. L'agente AgNOS prende le decisioni in base

all'ambiente, agli eventi generati dal network OS e dal contenuto dei messaggi che riceve. Un agente AgNOS può ricevere o inviare richieste. Può anche mandare informazioni riguardo al proprio stato interno. Inoltre può controllare le azioni di altri agenti se ha l'autorità per farlo. Ogni messaggio inviato è espressione di un effetto di una sua azione. Anche gli eventi sono un'altra fonte di informazioni. I controller possono non aver introdotto informazioni direttamente nella vista della rete. In questo caso gli agenti devono avere dei sensori capaci di gestire eventi, analizzandone il contenuto e aggiornando la propria knowledge base. I messaggi possono dare informazioni riguardo ad altri domini da altri agenti o contenere informazioni riguardo agli attributi di rete, spesso collegati a problemi associati ad altri domini. Un ambiente di un agente è una struttura dinamica, generata a secondo dello stato corrente della rete. Due cose sono necessarie per costruirlo: un linguaggio per rappresentare la conoscenza e un meccanismo di inferenza. Il linguaggio deve essere in grado di rappresentare schemi di ragionamento riguardo alle entità di rete e le loro relazioni. Le azioni che un agente AgNOS può fare sono quelle che ne cambiano lo stato o lo stato dell'ambiente, o quelle che permettono la cooperazione con altri agenti. Da qui in poi si parlerà di Information Resource per indicare i messaggi ricevuti dall'agente.

Definizione di Azione di un agente AgNOS: Sia Λ un insieme di stati possibili di un agente A, S un set di stati e Υ un set di information resource. Un'azione di A si definisce come: $A : \Lambda$ x S x $\Upsilon \rightarrow \Lambda$ x S x Υ

Da questa definizione si evince che quando un'azione non è sollevata da un messaggio, la terza coordinata è vuota. Un'azione quindi è la composizione di analisi dell'ambiente, decisione di cosa fare e esecuzione dell'azione, ossia generare un evento. Un agente AgNOS è capace di generare eventi che sono processati dal network controller, che mediante l'uso di direttive a basso-livello agli switch, modificano la vista della rete. Gli agenti AgNOS inoltre generano una specifica forma di azioni chiamati messaggi, che sono destinati a precisi agenti nel dominio. Questi messaggi possono essere visti come eventi.

AgNOS: ciclo di vita delle funzioni degli agenti

Si assuma che l'ambiente sia in uno stato istantaneo di un insieme finito E di eventi discreti. Gli agenti AgNOS hanno un range di possibili azioni, che ne trasformano lo stato. Un run r, di un agente su un ambiente, è una sequenza di stati ambiente e azioni:

Definizione di Run: Sia E un insieme finito di stati $\{e_0, e_1, ... , e_n\}$, e sia A un insieme finito di azioni $\{\alpha_0, \alpha_1, ..., \alpha_n\}$. Il run di un'agente si definisce come una sequenza:

$e_0 \xrightarrow{\alpha_0} e_1 \xrightarrow{\alpha_1} e_2 \xrightarrow{\alpha_2} e_3 \xrightarrow{\alpha_3} ... \xrightarrow{\alpha_n-1} e_n$

Gli stati ambiente di AgNOS rappresentano lo stato corrente della rete SDN. Ogni stato e_i indica come il dominio della rete è correntemente visto in termini di topologia e servizi. Le azioni degli agenti AgNOS sono limitati dal livello d'astrazione del controller di rete.

Definizione di agente AgNOS: Sia R_E un insieme di run che finiscono in uno stato ambiente, e A un insieme di azioni. Un agente AgNOS è definito come il mapping tra: $Ag_{\text{AgNOS}} : R_E \rightarrow A$

La comunicazione nell'AgNOS control plane è definito mediante un sottoinsieme di ACL (Agent Communication Language) [13]. L'obiettivo principale di ACL è fornire interoperabilità tra agenti di architettura diversa. Nel caso di AgNOS non c'è bisogno dato che gli agenti AgNOS hanno tutti la stessa architettura. Si utilizza quindi una versione più leggera di ACL.

Programmare con AgNOS

In questa sezione si illustrerà come definire un agente AgNOS e in che modo programmare le sue azioni.

Sintassi del linguaggio logico

Gli agenti logici AgNOS hanno uno stato interno di formule logiche che rappresentano la knowledge base. Ogni formula è rappresentata usando la notazione di clausole.

Definizione di formula atomica o letterale: Sia p un simbolo rappresentante una relazione tra i termini t_1, ... , tn. Allora una formula atomica è: $p(t_1, \ldots, t_n)$, oppure $\sim p(t_1, \ldots, t_n)$

Possiamo dire che un letterale è una formula atomica o la sua negazione. L e ∼L sono chiamate positive e negative letterali, e sono complementari.

Definizione di AgNOS Clausal Rules: Siano L_1, ... , L_n letterali. Una AgNOS Clausal Rule è definita come la loro disgiunzione $L_1 \vee \ldots \vee L_n$

Definizione di stato interno di AgNOS: Sia L un insieme di AgNOS Clausal Rules e sia $K=2^L$ un insieme di possibili stati della knowledge base. Se K è la composizione di KB_1, ... , KB_n, allora uno stato interno di un agent AgNOS è un elemento di K.

Ad esempio le seguenti frasi possono definire una knowledge base di un agente:

$superuser(alice)$.
$superuser(toddy)$.
$\sim attacker(alice)$.
$\sim attacker(toddy)$.
$tcp(6)$.
$udp(17)$.

$allow(\overrightarrow{Flow}) \vee \sim superuser(Us) \vee \sim udp(Prot) \vee \sim valid(Us)$
$valid(Us) \vee attacker(Us)$.

che stanno a significare che alice e toddy fanno parte del gruppo superuser e non sono degli attaccanti; le porte tcp e udp sono definite, le altre due regole definiscono rispettivamente le proprietà del flusso e che un attaccante non è un utente valido.

Le formule in una knowledge base di AgNOS esprimono la conoscenza riguardo la rete e gli altri agenti. Le formule eventi possono rappresentare eventi nel network controller. Le formule policy rappresentano le politiche definite nel linguaggio logico di AgNOS. Le formule messaggio rappresentano

interazioni basate su messaggi tra agenti.

Logical Reasoning Engine: Il motore di inferenza di AgNOS è un Clausal-Based Formal System [14] dove le clausole sono divise in due categorie: Clausole iniziali (o knowledge base, dette **B**) sono quelle che appartengono all'insieme degli assiomi più le negazioni delle query; Clausole derivate sono quelle prodotte tramite operazioni di inferenza. Se S è una query in forma clausale e B un insieme di clausole iniziali, allora la deduzione di S da B corrisponde a derivare una clausola vuota, $\sqcup$, da $\sim S \cup B$ o è uguale a provare che $\sim S \cup B$ è non soddisfacibile.

Definizione di Declarative Abstraction: Chiamiamo knowledge abstraction del livello dichiarativo le seguenti operazioni che permettono di immagazzinare e recuperare informazioni della knowledge. Indichiamo con α una query.

- prove(α): si chiede al motore di inferenza e viene tornata un'azione risultato della deduzione di α da KB (cioè chiediamo se $\{\sim \alpha\} \cup$ KB è soddisfacibile o no).
- knows(α): è un'azione che cambia la KB dell'agente da KB a KB $\cup \alpha$, dove α è della forma clausale.
- remove(α): è un'azione che cambia la KB dell'agente da KB a KB - α, α è clausale.

Usando queste tre astrazioni si può implementare il comportamento di AgNOS in ogni linguaggio imperativo. Nell'esperimento è stato scelto C++.

implementazione di AgNOS

L'implementazione di AgNOS estende e sviluppa il NOX network controller. Gli agenti intelligenti sono sviluppati in C++, come componenti NOX [15]. Gli agenti AgNOS formano un dominio SDN in grado di gestire un control plane. Questi agenti possono interagire dentro una singola organizzazione o interagire con altri domini SDN. L'interazione è gestita dal controller che

gestisce le priorità dei messaggi. L'aspetto principale del linguaggio utilizzato è la definizione di message types e message parameters. Message types denota le performance riguardo i messaggi inviati/ricevuti dagli agenti.

AgNOS: caso d'uso

DDoS (Distributed denial-of-service attacks) è una delle minacce maggiori contro gli ISP (Internet Service Providers). Molte tecniche sono state proposte [16, 17] per risolvere il problema e consentire alla vittima di droppare o limitare il traffico verso di essa, utilizzando filtri che bloccano l'eccessivo afflusso di pacchetti. La natura distribuita di un attacco DDoS complica le cose in uno sviluppo su larga scala di queste architetture, dato che ogni AS dovrebbe aggiornare gli end-system o i router al fine di offrire il servizio garantito.

Percezione dell'ambiente

La percezione dell'ambiente da parte degli agenti AgNOS può essere di due tipi: eventi o messaggi. Ci sono eventi generati direttamente dal network controller e eventi generati da applicazioni di gestioni ausiliarie, come gli Authenticator. Gli agenti AgNOS non gestiscono direttamente gli eventi, ma aggiornano la loro knowledge base in relazione all'effetto provocato sull'ambiente da un evento. Gli eventi che vengono gestiti direttamente dagli agenti AgNOS sono quelli generati da un 'attack detector'. Un attack detector [18] è un componente col compito di analizzare il traffico di rete e decidere se la rete è sotto attacco DDoS.Il detector usa una map auto-organizzata che identifica pattern di attacchi sul traffico. Non analizza pacchetto per pacchetto, ma usa informazioni di flusso disponibili da OpenFlow negli switch, forniti dal network controller. I messaggi sono inoltre gestiti direttamente dagli agenti AgNOS. E' importante osservare che i messaggi possono essere ricevuti dagli agenti in ogni momento, sia che il receiver sia un attaccante che una vittima di un attacco.

Processo decisionale di AgNOS

Il ciclo di vita di un agente AgNOS prevede che venga aggiornato il proprio KB per ogni percezione dalla rete. Inizialmente ogni KB è data dalle policy del network domain controllato. Una volta che un agente rileva un attacco DDoS, prende decisioni per mitigare l'attacco utilizzando il meta-interprete descritto nella sezione III.

Azioni sull'ambiente

Il framework prevede che l'agente agisca in due modi:

- Generando eventi che sono gestiti dall'applicazione di gestione o dal network controller
- Mandando messaggi agli altri agenti AgNOS

Dal punto di vista della rete attaccata, l'azione migliore è quella che chiede l'origine dell'attacco al fine di bloccare il flusso o limitarlo. In questo caso AgNOS genera eventi messaggio. Viene fatta una request e vengono informati i peers vicini che si è sotto attacco DDoS. Sia inform che request arrivano a destinazione sotto forma di eventi, al fine di aggiornare il KB degli agenti nel dominio attacco. Il meccanismo autonomo di questo agente permette di scegliere autonomamente di cooperare con gli altri agenti al fine di bloccare l'attacco. Se decide di bloccare il flusso all'origine, genera un flow_mod_events o manipola direttamente la vista della rete al fine di bloccare il flusso nel datapath utilizzando il network controller. Dopo averlo bloccato è l'attack detector a decidere se il traffico è accettabile e genera un evento attack mitigated per far aggiornare il KB di ogni agente.

Esperimento

Nell'esperimento si è usata una rete basata su Mininet [8, 9]. Si è costruito una rete SDN per testare una DDoS mitigation.

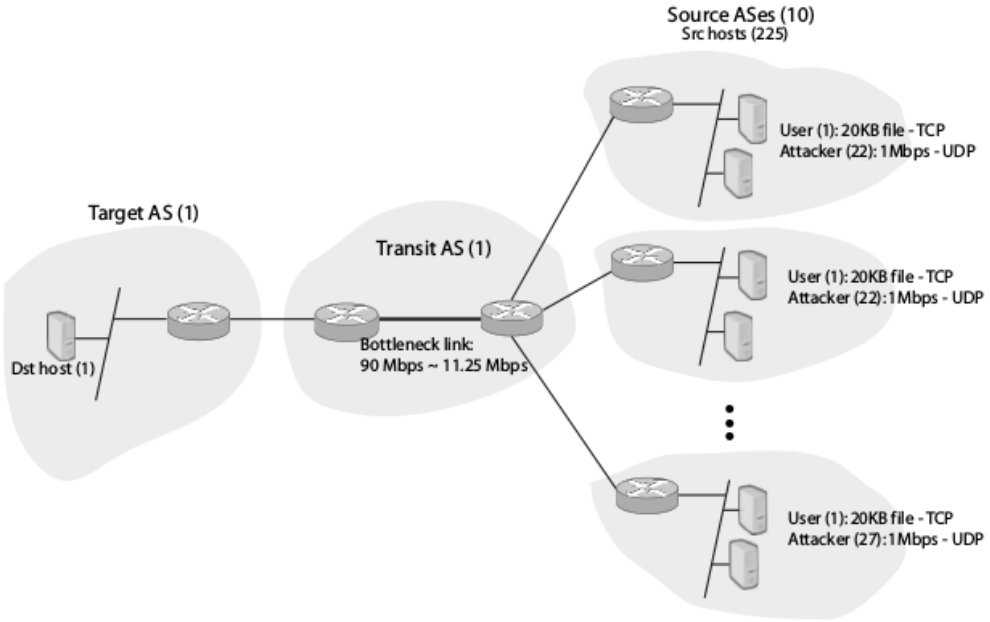

Figura 3.3: Topologia per lo scenario DDoS [5]

Gli autori [5] vogliono simulare gli attacchi da migliaia a milioni di attacchi verso un dato link. Purtroppo la richiesta troppo alta di risorse ha richiesto di ridimensionare la topologia riducendola ad alcune migliaia di nodi. Hanno deciso però di seguire il seguente approccio: Fissato il numero di nodi, si riduce la capacità del link creando un collo di bottiglia, al fine di simulare uno scenario più grande, aumentando man mano il numero di attaccanti. Nella topologia semplificata si sono utilizzati 10 AS connessi all'AS di destinazione mediante un AS di transito. Ogni AS ha 100 hosts connessi a un singolo access router. Il transit AS ha due router e il Target AS ha un unico host vittima. Ogni AS esegue Mininet, dove ogni router è emulato con switch virtuali OpenFlow. Ogni host è un Host Mininet-based (macchina Linux); per ogni AS c'è un network controller con un sistema AgNOS. Il link nell'AS Transit è un collo di bottiglia e tutti gli altri link hanno sufficiente capacità per evitare la congestione. Gli autori hanno variato la capacità del link collo di bottiglia da 90Mbps a 11.25Mbps per emulare uno scenario dove 25.000

$\sim$ 200.000 senders condividono un canale da 10Gbps. Ogni sender condivide una bandwidth da 400Kbps a 50Kbps. Il ritardo di propagazione di ogni link è 10ms. Per fare uno stress test, ogni source AS ha solo un utente legittimo che ripetutamente invia 20KB alla vittima usando il protocollo TCP. Ogni attaccante invia un traffico UDP costante da 1Mbps alla vittima.

Valutazione delle Performance

Gli autori vogliono comparare le performance di AgNOS con le più diffuse architetture DDoS filter-based. Gli obiettivi sono due:

- Dimostrare che gli agenti AgNOS funzionano nella rete
- Mostrare come le loro performance siano comparabili a quelle già affermate in letteratura.

Come architettura filter-based hanno scelto StopIT [19] e Fair Queuing (FQ) [16]. StopIT è un sistema defense filter-based. Con questo approccio la vittima può installare filtri di rete per fermare il traffico attaccante. Il Fair Queuing 'stringe' il traffico attaccante al fine di ridurlo a un traffico 'fair'.

Due scenari sperimentati: un esperimento con 'Unwanted Traffic Flooding Attacks' dove gli attaccanti colpiscono direttamente la vittima con un flusso, ma la vittima può classificare l'attacco e usare i meccanismi di difesa implementati in AgNOS; un secondo esperimento con 'Colluding Attacks' dove la vittima riceve un attacco indiretto rivolto verso un link intermedio.

Analisi dei risultati

L'obiettivo dell'agente è mitigare l'attacco e mantenere il servizio funzionante. La figura mostra il tempo di trasferimento medio di 20KB da un host legittimo verso l'host destinatario sotto attacco.

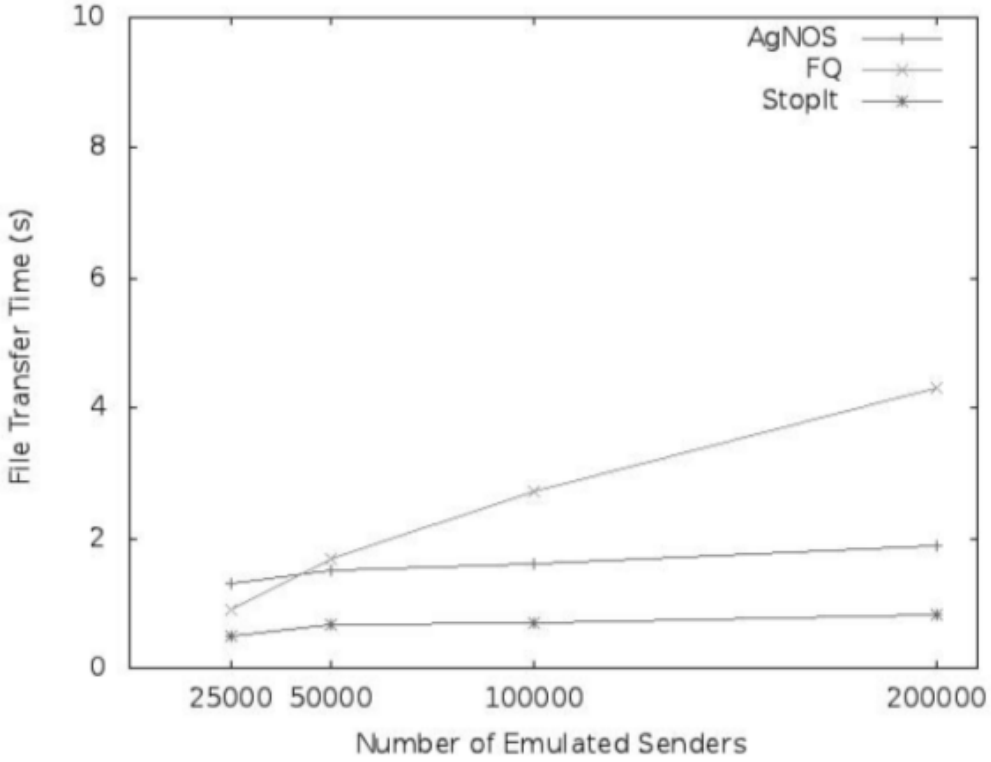

Figura 3.4: Tempo medio di trasferimento durante l'attacco DDoS [5]

Nel grafico sono stati inseriti anche i risultati raggiunti tramite l'architettura StopIt e il meccanismo Fair Queuing. La topologia e metodologia degli esperimenti sono le stesse per tutte le architetture. Il risultato è relativo al primo esperimento concernente il 'Unwanted traffic flooding attacks'.

La figura mostra come Fair Queuing sia quella con performance peggiori, perché aumenta linearmente con il numero di senders emulati quando i pacchetti competono con il traffico attaccante nel collo di bottiglia. StopIT invece ha le performance migliori. Per quanto riguarda l'architettura AgNOS fornisce un accettabile tempo di trasferimento medio. Definiscono accettabile il tempo minore a quello di FQ per la stessa configurazione. Con AgNOS il tempo cresce di un fattore costante con l'aumentare degli attaccanti. Inoltre il grafico mostra come il servizio rimanga attivo sull'host vittima. C'è un'importante differenza tra StopIT e AgNOS: il primo implementa una policy

di traffico robusta che controlla i loop che garantisce a ogni sender un 'fair share' della bandwidth mentre AgNOS non prevede questo controllo; AgNOS invece prova a bloccare il traffico il prima possibile colpendolo alla fonte. Gli autori non considerano questo come un limite dato che le reti SDN possono facilmente limitare i flussi di dati.

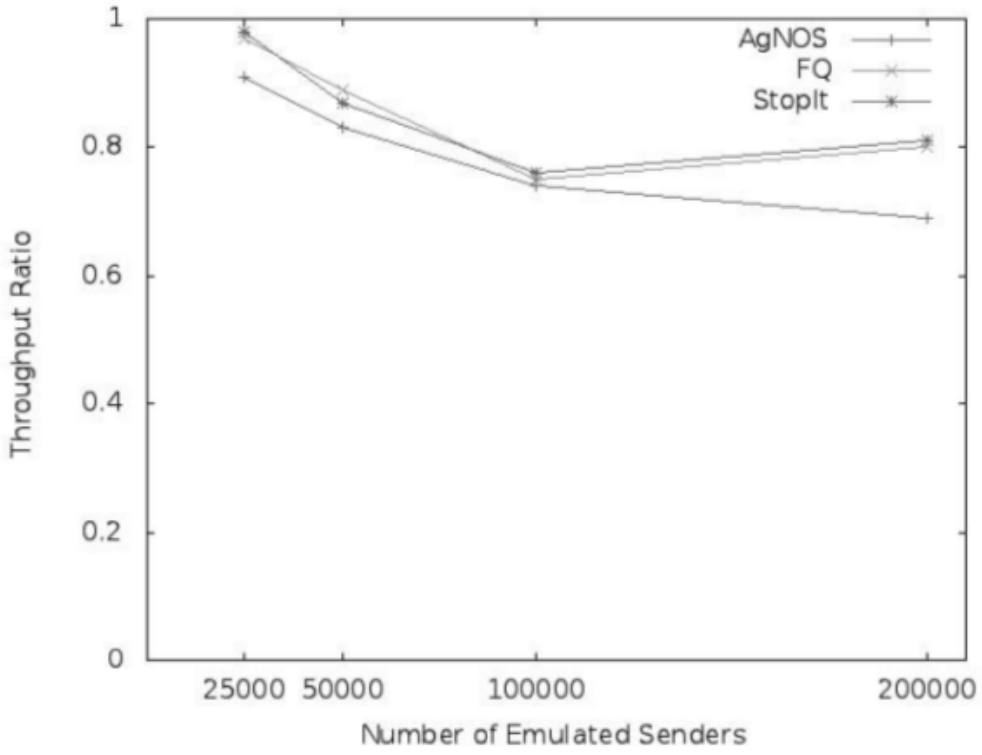

Figura 3.5: Throughput ratio tra attacco e traffico normale [5]

La figura mostra il throughput ratio tra l'attacco e il traffico normale in uno scenario congestionato. Lo scopo del calcolo del rapporto è capire se un utente legittimo riceve lo stesso throughput medio nel collo di bottiglia come un nodo malevolo. FQ e StopIT si comportano in modo simile perché usano una coda per-sender fair per proteggere il traffico legittimo. L'architettura AgNOS si comporta similmente a FQ/StopIT eccetto per 200.000 attaccanti diminuendo il throughput ratio. Rimane un meccanismo accettabile dato che il servizio non smette di funzionare nel target host. AgNOS continua

a mandare messaggi agli altri domini che chiedono di bloccare i pacchetti e li informa dell'attacco. Questo attacco DDoS stressa la necessità di avere software auto-regolati che identificano velocemente pattern di attacchi prima che la vittima risulti inaccessibile.

Conclusioni

Negli esperimenti, gli autori hanno ottenuto migliori prestazioni di quelle ottenute dai sistemi di reti autonomi che utilizzano analisi pacchetto per pacchetto. E' inoltre difficile fare una comparazione oggettiva dato che molti AS autonomi sono solo teorizzati, mentre AgNOS è realmente implementato. Un'altra differenza critica è l'esecuzione degli agenti nell'ambiente; gli esperimenti fatti sono i primi a implementare agenti intelligenti nello stesso ambiente del network controller. Gli agenti funzionano come applicazioni di gestione di rete e sono serviti direttamente dalla vista della rete. Sviluppare agenti intelligenti per ogni switch è una scelta da evitare: Internet ha più di 40.000 AS. Gli autori credono che la possibilità di sviluppare agenti intelligenti sui network controller può motivare i gestori di rete ad adottare soluzioni cooperative su larga scala. Con un risparmio considerevole si otterrebbero maggiori performance aggiornando solamente il software nei network controller centralizzati.

3.2 OpenADN

In questa sezione analizzeremo l'esperimento condotto dalla Washington University, che ha cercato di capire la compatibilità di SDN a operare in un ambiente multi-cloud [7].

Introduzione

Virtualizzazione è la chiave del successo corrente e futuro del cloud computing. Internet ha portato alla virtualizzazione di tutti gli aspetti della vita,

dal comprare online, all'apprendimento, all'intrattenimento.

Perché virtualizzare? Condivisione: Risorsa troppo grossa sfruttata solo da un utente o idem con multiprocessori o linee ad alta velocità o dischi ad alta capacità. Isolamento: La virtualizzazione permette di dividere logicamente l'uso delle risorse fisiche, consentendo di usufruirne senza interferire tra un'attività e l'altra. Aggregazione: Quando una risorsa è troppo piccola se ne può costruire una più grossa logica come insieme di più risorse piccole. Ad esempio più hard disk inaffidabili sfruttati per server farm affidabili. Dinamicità: Risorse cambiano velocemente e hanno bisogno di essere riallocate in modo rapido. Più facile se sono virtuali e non fisiche. Facilità di gestione (più importante): Device virtuali facili da gestire perché sono software-based e hanno interfacce spesso standard

La virtualizzazione non è un concetto nuovo per quanto riguarda le reti, ad esempio i canali virtuali con reti X.25 che permettevano di condividere un canale fisico grande tra più utenti. VLAN (Vitual local area networks) permette alle aziende di condividere LAN fisica con riservatezza. VPN (Virtual private networks) permette alle aziende di usare reti pubbliche in sicurezza. Il cloud computing ha trascinato l'ambiente informatico a sviluppare sempre nuovi standard per quanto riguarda la virtualizzazione di rete, SDN ha un ruolo chiave nello sviluppo della rete in questo senso.
Per far fronte alle esigenze della rete molti componenti sono stati virtualizzati come ad esempio i NIC (Network Interface Card) e gli switch per permettere al numero sempre crescente di host in rete di essere connesso non dovendo per forza usare una quantità spropositata di switch fisici.
Un altro problema è quello della migrazione delle macchine virtuali, che se passano da una sottorete ad un'altra devono cambiare indirizzo IP complicando così routing. L'indirizzo IP identifica e localizza un sistema, è molto più facile migrare sistemi in una sottorete, questo perché gli indirizzi usati nelle sottoreti(livello 2) sono solo identificatori e non localizzatori del sistema e non cambiano quindi quando un sistema migra. Per questo quando due sottoreti sono collegate attraverso una rete di livello 3 è desiderabile costruire

una rete virtuale di livello 2 che comprende l'intera rete in modo da rendere agevole lo spostamento di macchine virtuali.
Nel cloud computing la gestione multipla di più macchine virtuali su una singola macchina fisica potrebbero appartenere a client diversi su vLAN diverse, ogni vLAN potrebbe estendersi su diversi data center interconnessi via rete di livello 3. Ci sono diverse proposte di soluzione tra cui quella di una Virtual extensible LAN (VXLAN) [21].
Una società con più data center localizzati in parti diverse potrebbe voler spostare le proprie macchine virtuali ovunque in modo rapido e semplice. Si potrebbe volere quindi che tutte le macchine siano connesse ad una singola Ethernet virtuale, per farlo si può usare un Medium Access Control (MAC) over IP.
L'aumentare di tutti questi dispositivi virtuali ha creato la necessità di poterli spostare e gestire in modo semplice e adeguato. Proprio per questo si è capito che SDN potrebbe rappresentare una soluzione in quanto permetterebbe di orchestrare un largo numero di dispositivi, fisici e virtuali, simultaneamente. Si possono così gestire policy e flussi di traffico a seconda del tipo di applicazioni, o dal contesto di applicazione, dall'utente o dal contesto server o dai requisiti QoS dell'applicazione. Il servizio applicazione può essere replicato su più host. Inoltre il servizio può essere partizionato per migliorarne le performance, dove ogni partizione è in esecuzione su un gruppo di server diverso. La partizione può essere basata su:

- Contenuto: Ad esempio per lo stesso servizio (es. videos.google.com), la gestione dei messaggi o le richieste o le recommendation possono essere gestiti da server diversi

- Contesto: Che può essere il contesto utente, il contesto di rete o il contesto server; questa divisione può portare l'applicazione a fare un routing diverso dei messaggi.
  Un esempio di contesto utente è dato dal dover trattare in modo diverso gli utenti mobile da quelli desktop.
  Un esempio di contesto di rete, invece, si ha in base alla diversa loca-

zione geografica dei vari utenti.
Un esempio di contesto server è rappresentato dal carico di sistema dei vari server appunto e dei periodi in cui questi sono attivi oppure no.

Inoltre, molti servizi richiedono che più segmenti TCP debbano passare prima da una serie di device intermedi come firewall o IDS per motivi di sicurezza, o da ottimizzatori di performance come ad esempio SSL off loaders o WAN optimizers. Di solito una connessione user-server non è end-to-end, ma è formata da più segmenti. Ogni segmento può essere gestito da destinatari diversi, in base alla replicazione o al partizionamento. Per questo gli Application Service Providers (ASP) implementano sofisticate politiche di instradamento nei loro data center privati.

Problema

Molte applicazioni al giorno d'oggi (come i giochi per smartphone) hanno bisogno di servire utenti di tanti paesi diversi, per questo necessitano di server distribuiti nell'intero globo. Questo può essere fatto facilmente servendosi dei servizi di cloud di molteplici provider sparsi per tutto il mondo. Il problema è che in questo contesto multi-cloud molto dinamico non è possibile usare le politiche di instradamento degli ASP in quanto gli ISP non offrono nessun servizio che permetta l'instradamento dinamico dei pacchetti a server diversi.

Approccio alla soluzione

La soluzione proposta [7] è la progettazione di un nuovo livello di astrazione di sessione chiamato Open Application Delivery Network (OpenADN) [22] che permette agli ASP di spedire e migliorare il traffico gestendo in modo granulare i messaggi e i pacchetti delle applicazioni. Permette agli ASP di sfruttare le sofisticate politiche di instradamento dei pacchetti utilizzate nei loro data center privati ma in scala globale multi-cloud. OpenADN si basa sull'architettura SDN, in questo modo gli ISP possono offrire i servizi di instradamento di cui gli ASP hanno bisogno. La realizzazione di

OpenADN è possibile combinando 6 innovazioni: OpenFlow, SDN, session splicing, cross-layer communication, indirection, MPLS-like application flow labels.

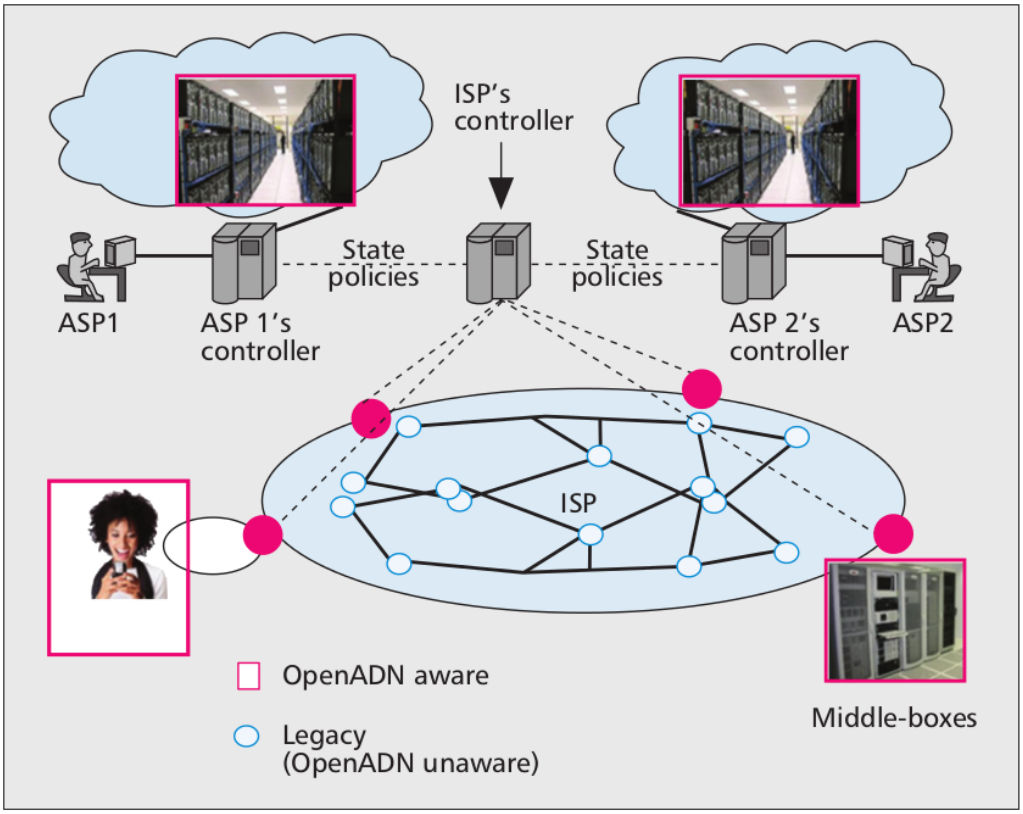

Figura 3.6: OpenADN [7]

OpenADN permette ai controller ASP di comunicare con i controller ISP e fornisce a quest'ultimi le policy e lo stato del server così che si possa programmare il piano di controllo in modo adeguato.

Punti chiavi di OpenADN

- OpenADN porta all'estremo la virtualizzazione della rete facendo apparire Internet come un singolo data center virtuale a ogni ASP.
- I Proxy possono essere posizionati dappertutto. Più sono vicini agli utenti e ai server e più sono performanti.

- Retrocompatibilità con il traffico standard
- Non c'è bisogno di cambiare il core di Internet dato che solo alcuni collegamenti tra devices hanno bisogno di OpenADN/SDN/OpenFlow. I devices e router rimanenti possono rimanere come prima.
- Gli ISP potrebbero finanziare l'utilizzo di OpenADN beneficiando subito chi ne fa uso.
- Gli ISP mantengono il controllo sulle risorse di rete mentre gli ASP hanno il controllo sui dati delle loro applicazioni, che potrebbero essere confidenziali e criptati.

3.3 Procera

In questa sezione si parlerà degli esperimenti condotti dalla Georgia Institute of Technology [4], che descrivono diversi prototipi di rete, per il campus universitario e per reti domestiche servendosi di SDN, dimostrando come l'uso di questa architettura possa migliorare sensibilmente la gestione della rete.

Introduzione

Gli amministratori di rete hanno bisogno di implementare politiche sempre più sofisticate che si traducono nel dover sviluppare funzioni sempre più complesse, dovendo però lavorare con un ristretto insieme di istruzioni di basso livello in un ambiente a linea di comando. Ma non solo, non hanno nemmeno gli strumenti per potersi adattare in tempo reale ai mutamenti dello stato della rete, che cambia in continuazione; gli operatori di rete infatti devono modificare manualmente le configurazioni di rete ogni volta che questa subisce un cambiamento. Per questo vengono costruiti script ad-hoc per riconfigurare le rete in modo dinamico, portando così a dover fare frequenti cambiamenti aumentando il rischio di inserire errori [23].

In risposta a questo SDN fornisce i mezzi per una corretta e semplice gestione della rete. Per fornire agli amministratori di rete un modo semplice per implementare le sofisticate politiche di alto livello la Georgia Institute of Technology ha deciso di basarsi sul paradigma SDN, sviluppando Procera [25], un framework ad eventi per il controllo della rete. Il linguaggio per le policy e Procera si basano sul functional reactive programming (FRP). Procera permette agli amministratori di rete di esprimere le policy di alto livello con questo linguaggio e traduce queste policy in una serie di regole di forwarding da applicare all'infrastruttura sottostante usando OpenFlow. Si è usato Procera per rimplementare le policy di rete esistenti al Georgia Tech campus, che usa complicate VLAN e molte middleboxes per implementarle. Inoltre usando anche BISmark [24] è stato implementato un sistema di gestione di rete casalinga.
Lo scopo è quello di dimostrare come Procera tramite SDN possa notevolmente ridurre il lavoro di gestione e introdurre nuove funzionalità in maniera molto semplice.

Procera

Procera è un framework per il controllo di rete che aiuta gli operatori a esprimere policy di rete basate su eventi che reagiscono in modo differente ai vari tipi di eventi che possono essere sollevati usando un linguaggio di programmazione funzionale di alto livello. Procera fa da collante tra le policy ad eventi di alto livello e le configurazioni di rete a basso livello. Per esprimere queste policy di rete Procera offre un set di *control domain* che gli amministratori possono usare per impostare determinate condizioni ed assegnare regole di forwarding adeguate corrispondenti a tali condizioni. Gli operatori di rete posso anche combinare vari *control domain* per costruire sofisticate politiche di rete. L'insieme dei *control domain* supportati da Procera sono illustrati nella seguente tabella:

Control domains	Examples
Time	Peak traffic hours, academic semester start date
Data usage	Amount of data usage, traffic rate
Status	Identity of device/user, distinct policy group, authentication status
Flow	Ingress port, ether src/dst, ether type, vlan id, vlan priority, IP src/dst, IP protocol, IP ToS bits, port number src/dst

Figura 3.7: Control domain di Procera [4]

Time: In molti casi si ha bisogno che il comportamento della rete sia diversificato in base alla data o all'ora del giorno. Per esempio un amministratore di rete di un campus universitario potrebbe voler gestire il traffico in modo diverso durante l'anno accademico rispetto ai periodi di pausa dove l'utenza è molto minore. In una rete domestica gli utenti potrebbero voler impostare dei filtri sui contenuti in base all'orario.

Data usage: Si potrebbe voler specificare policy diverse laddove il comportamento della rete dipenda dall'ammontare del traffico dati (download/upload) o dalla velocità di trasferimento dati in un particolare intervallo di tempo.

Status: Gli amministratori potrebbero voler specificare privilegi diversi per differenti utenti o gruppi di essi. Inoltre i privilegi legati ad un utente possono cambiare per vari motivi. I privilegi di un device vanno cambiati in base all'utente che sta usando quello specifico device.

Flow: Si ha bisogno che la reti si comporti in modo diverso anche in base a vari campi dei diversi livelli specificati nei pacchetti o nei flussi. Un flusso, flow, è una 12-tupla specificata da OpenFlow.

La seguente figura mostra l'architettura di Procera che successivamente verrà spiegata nel dettaglio.

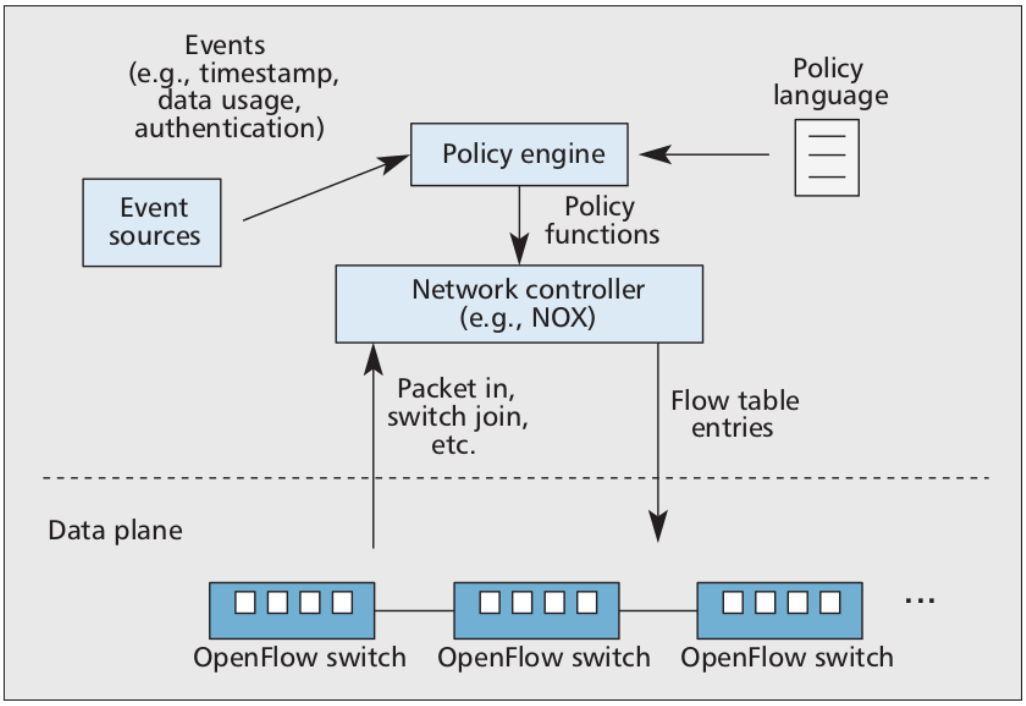

Figura 3.8: Architettura di Procera [4]

Event sources

Gli *event source* possono essere tutti quei dispositivi che possono sollevare dinamicamente degli eventi rilevanti per una rete come Intrusion Detection System, Network Bandwidth Monitoring System, Authentication System, Simple Network Managment Protocol (SNMP), i valori in /proc e qualsiasi altro elemento che possa sollevare eventi. Nel *policy engine* c'è un parser che riesce a comprendere questi eventi. Non è definita un interfaccia fissa tra il *policy engine* e le fonti degli eventi, ci possono essere diversi metodi come per esempio le JSON-RPC.

Policy engine e Language

Il *policy engine* deve parsare le policy di rete espresse nel *policy language* e deve processare tutti gli eventi che gli pervengono dagli *event sources*. Basandosi sul *policy language* e sugli eventi che il *policy engine* riceve in modo

asincrono, esso aggiorna il proprio stato, definendo in questo modo quale politica di rete debba essere messa in atto e invia le adeguate funzioni al controller di rete. Alcune policy cambiano lo stato semplicemente al variare dell'ora del giorno senza ricevere alcun evento esterno, è il *policy language* a supportare questi tipi di cambiamenti.
Il *policy language* di Procera è basato sul FRP, questo permette agli amministratori di specificare sofisticate policy in un semplice linguaggio dichiarativo.

Controller

Procera si basa sul paradigma SDN, che quindi ha un controller che prende tutte le decisioni per l'inoltro di pacchetti aggiornando le flow-table degli switch del livello sottostante in base alle politiche di rete da seguire. Il controller stabilisce una connessione con ogni switch OpenFlow e inserisce, cancella o modifica le regole di forwarding negli switch usando il protocollo di OpenFlow. Inoltre il controller reagisce agli eventi packet_in e switch_join che vengono dagli switch, per i primi il controller istallerà nuove regole di forward nello switch, mentre per i secondi il controller stabilirà una connessione con quello specifico switch. Attualmente Procera usa le specifiche di OpenFlow della versione 1.0.0.

Rete del campus

Descriviamo lo sviluppo di Procera in una rete di un campus. Solitamente le reti dei campus sono dinamiche e caratterizzate da numerosi eventi; similmente alle reti aziendali, queste reti sono molto complesse e facilmente esposte a rischi, quindi particolarmente adatte per lo sviluppo di Procera. La rete della Georgia Tech richiede a ogni host non registrato, l'autenticazione tramite un portale web; dopo essersi autenticati con successo mediante un nome utente ed a una password, il device in uso viene sottoposto ad una scansione di vulnerabilità; se non vi sono rischi, il device viene connesso alla rete Internet. Questa versione semplicistica del meccanismo nasconde una procedura complessa che coinvolge tool esterni; in particolare la rete della

Georgia Tech si basa su virtual LAN (VLAN), dove device registrati e non, sono separati da differenti domini VLAN. Ogniqualvolta un device si autentica, cambia di dominio, gli switch devono quindi scaricare ogni volta la VLAN map aggiornata dal VLAN management server (VMPS) per permettere il corretto forwarding dei pacchetti.
Implementare un meccanismo simile con tool statici come regole di firewall e tecnologie VLAN, richiede agli operatori di rete configurazioni indipendenti per ogni componente diversa e numerosi script ad hoc. Procera semplifica enormemente queste tipologie di policy.

Policy

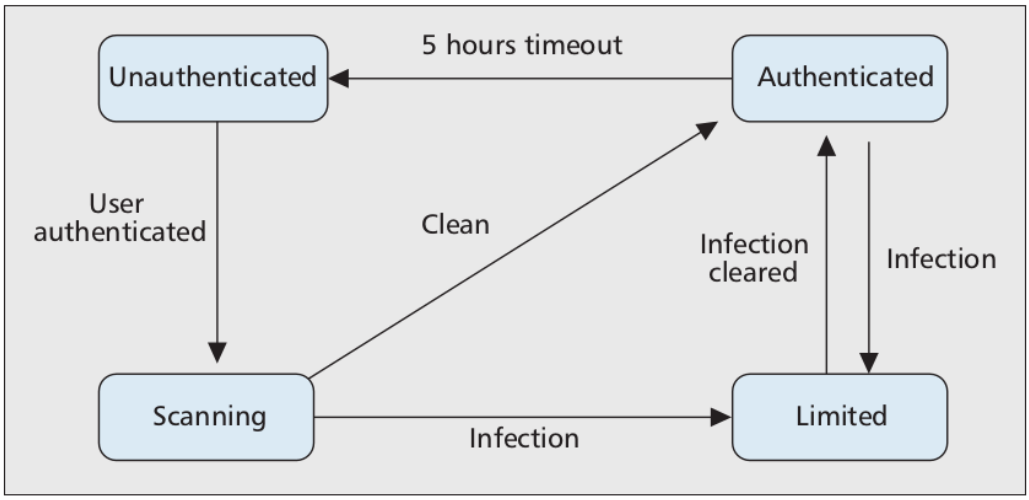

Figura 3.9: Transizioni ed eventi nella rete del campus [4]

La figura mostra la policy della rete della Georgia Tech come un automa a stati finiti. La policy è espressa mediante eventi e transizioni tra stati differenti. I device utente nel dominio Unauthenticated non hanno accesso alla rete. Un'autenticazione avvenuta con successo (username e password), permette il passaggio nello stato Scanning, dove è consentito solo il transito di pacchetti per scansionare le eventuali vulnerabilità nel dispositivo, se nessuna vulnerabilità è stata rilevata si passa nello stato Authenticated dove è consentito l'accesso a Internet. In qualsiasi momento venga sollevato un

evento da un IDS per qualche infezione, sia che ci si trovi nello stato Scanning che il quello Authenticated, si passa nello stato Limited e Internet viene bloccato. Dopo 5 ore di inattività, l'utente viene disconnesso e riportato nello stato Unauthenticated.

Deployment status

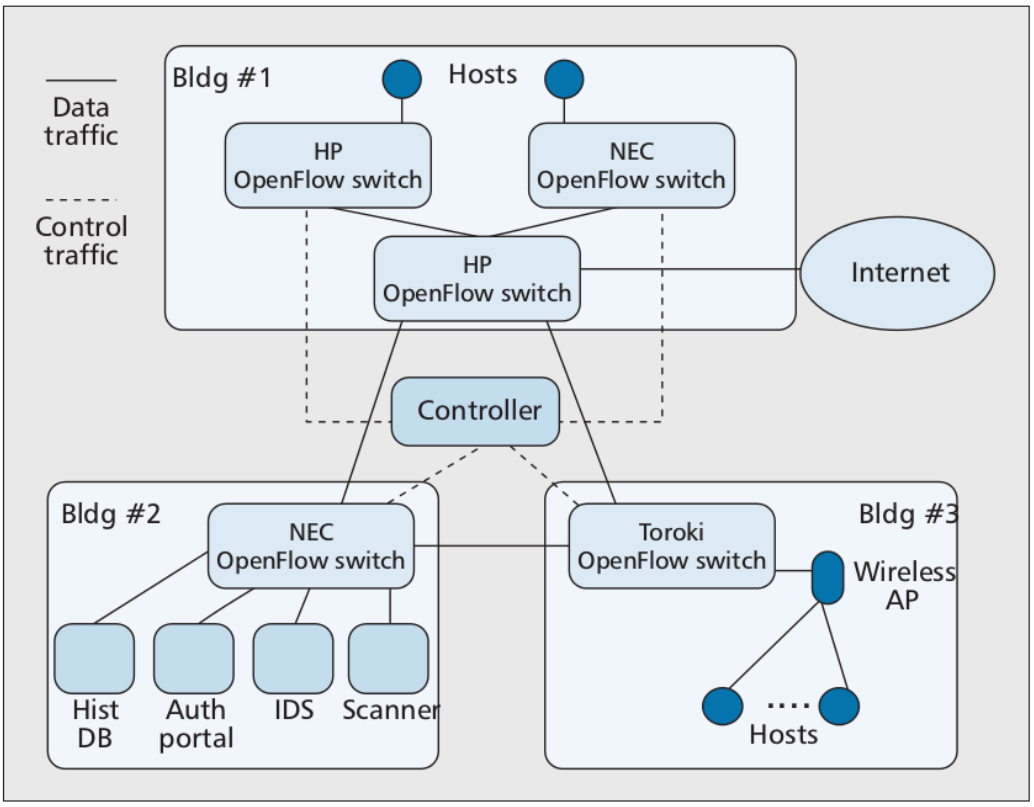

Figura 3.10: Struttura della rete del campus [4]

La figura mostra come viene sviluppata la rete tra i tre edifici del Georgia Tech campus. Per il forwarding dei pacchetti vengono utilizzati switch HP, NEC e Toroki con tecnologica OpenFlow. Vi sono due access point wireless nell'edificio 3, attraverso il quale gli host possono connettersi tramite SSID broadcasted. Il portale di autenticazione, il sistema di identificazione di intrusioni e lo scanner, sono situati nel data center nell'edificio 2.

Rete domestica

Si descrive lo sviluppo di Procera per una rete casalinga e la facilità con cui si possono esprimere le varie tipologie di policy usando questo framework.

BISmark: più visibilità

Uno dei problemi delle reti domestiche è che queste offrono una limitata visibilità per quanto riguarda le performance e lo stato. Per misurarne la velocità, l'utente è costretto a utilizzare tool web come speedtest.net che fornisce risultati discreti, influenzati da molti fattori come il browser o la condizione dell'host utilizzato. Gli Internet service provider (ISP) spesso vogliono monitorare costantemente le reti domestiche, garantendo il servizio offerto. I fornitori di contenuti vogliono inoltre conoscere l'esperienza utente per dirigere correttamente il proprio traffico. BISmark è un insieme di gateway installati in ambito domestico, un sistema di gestione centralizzato, un server dati e più server utilizzati per misurare le prestazioni. I gateway casalinghi svolgono diverse tipologie di misurazioni, attive e passive, elaborate dai sistemi centralizzati. Nel Novembre 2012 si contavano 270 gateway BISmark attivi in tutto il mondo.

SDN: più controllo

A causa dei sistemi chiusi installati nei gateway comuni casalinghi, è estremamente difficile introdurre nuove funzionalità di rete. SDN rende questo facile e possibile. É possibile combinare le misurazioni dati di BISmark e Procera per costruire un sistema di gestione capace di adattarsi a numerose condizioni diverse di rete. Per esempio è possibile modellare il traffico basandosi sulle richieste di performance o attuare meccanismi di prefetching e caching di contenuti di Internet. Usando SDN si possono fare varie decisioni sulla gestione del traffico e mettere queste regole nel gateway casalingo, in questo modo si accresce molto la flessibilità delle reti domestiche.

Caso d'uso in una rete domestica: uCap

Si dimostra come i gateway basati su OpenFlow, insieme alla suite BISmark, permette il monitoraggio delle reti domestiche e permette di controllare il traffico dati effettuato dai vari device connessi. Gli ISP sono soliti limitare mensilmente l'uso possibile del traffico dati [26]. Sfortunatamente gli utenti non hanno strumenti facili per monitorare l'uso che fanno in rete, nonostante alcune interfacce web fornite dagli stessi ISP che forniscono una visione generale e non particolare dei singoli device. Gli utenti di reti domestiche hanno bisogno di un sistema e di un'interfaccia che permetta non solo di monitorare l'uso del traffico dati per ogni device, ma anche di poter gestire l'uso della banda e porre dei limiti ad ogni singolo device; disattivando i dispositivi che raggiungono la soglia(cap) se l'utente vuole. Procera permette l'implementazione di queste regole sui device senza dover mettere mano alla configurazione di rete a basso livello tramite script ad hoc. Il paradigma SDN si adatta bene al sistema dato che la logica è implementata nei server back-end, come router casalinghi.

Policy

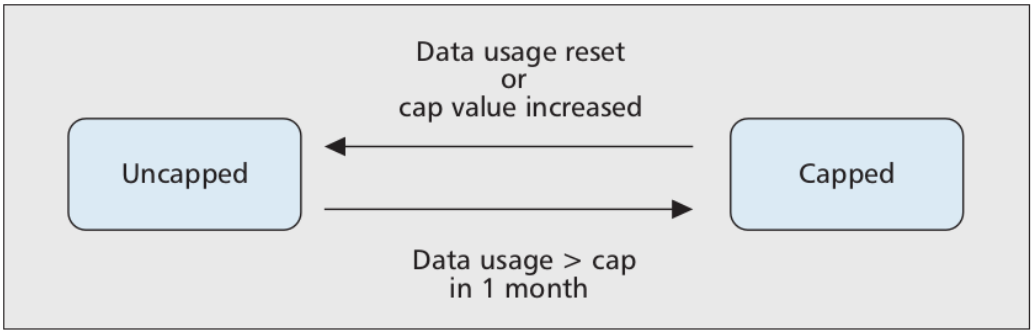

Figura 3.11: Stati e transizioni in una rete casalinga [4]

La figura mostra lo sviluppo delle policy in una rete domestica. Quando un device è Uncapped, un utente può connettersi a Internet normalmente, altrimenti viene bloccato se é Capped. La transizione tra Uncapped a Capped avviene quando l'utente supera il limite (cap) mensile d'uso della rete,

stabilito dall'amministratore di rete. Il passaggio inverso avviene mediante un reset (nuovo mese) o se il limite mensile viene alzato, ripristinando l'uso precedentemente bloccato. Gli eventi vengono gestiti dai router wireless, che inviano un report dell'uso della rete al server back-end ogni 5 secondi. Procera automaticamente individua ogni device capped o uncapped e configura le regole di forwarding adeguate nel router.

Conclusioni

Per semplificare i vari aspetti di gestione di rete, è stato sviluppato Procera, un framework di controllo basato su SDN. Gli operatori interagiscono con 4 domini di controllo: tempo, data usage, status di autenticazione e flusso di traffico al fine di implementare un sistema di policy reattive di rete con un linguaggio di configurazione ad alto livello basato sulla programmazione funzionale reattiva. Si è usato il protocollo OpenFlow per comunicare tra il controller Procera e gli switch sottostanti. Si è testato il tutto sia in una rete campus che in una rete domestica dimostrando come Procera sia praticabile e funzionale, permettendo di adempiere a gli obiettivi prefissi riducendo la complessità di gestione.

Capitolo 4

Conclusioni

É ormai noto che l'attuale architettura di Internet sia inadeguata per soddisfare tutti i requisiti attuali degli utenti e degli amministratori di rete. Come denotato dalle ricerche invece l'architettura SDN offre una grande innovazione separando il piano dati da quello di controllo ed aggiungendo un livello di astrazione superiore. Questo dà la possibilità di programmare il piano di controllo in un linguaggio di alto livello e rende la rete estremamente flessibile e di facile gestione. Come evidenziato dal caso d'uso di Procera SDN rende il lavoro di gestione delle reti molto più semplice ed evita che siano introdotti molti errori umani. Si è anche notato, grazie al caso d'uso di OpenADN, come SDN si adatti perfettamente ad ambienti multi-cloud aumentandone la flessibilità di cui questi hanno bisogno. Si è inoltre dimostrata l'efficacia di basare un framework ad agenti su SDN per cercare di limitare il DDoS.
La limitazione maggiore di SDN è che non ci sono ancora API standard per quanto riguarda la comunicazione tra application plane e control plane, questo rende lo sviluppo di applicazioni per SDN più difficoltoso. Per raggiungere il massimo potenziale di questa architettura si ha bisogno quindi di trovare le giuste API open source e si deve avere una diffusione in tutta la rete.
Visti gli enormi punti di forza questa nuova tecnologia si sta diffondendo velo-

cemente e continuerà a farlo, sempre nuove applicazioni verranno sviluppate basandosi su questo modello, proprio per questo il Software Defined Network sembra davvero essere il futuro della rete.

Bibliografia

[1] Open Networking Foundation: "Software-Defined Networking: The New Norm for Networks", ONF White Paper, Aprile 13,2012

[2] Open Networking Foundation: "SDN Architecture Overview", Version 1.0, Dicembre 12,2013

[3] Open Networking Foundation: "OpenFlow-enabled SDN and Network Functions Virtualization", ONF Solution Brief, Febbraio 17,2014

[4] H. Kim, N. Feamster, Georgia Institute of Technology: "Improving Network Management with Software Defined Networking", IEEE Communications Magazine, Febbraio 2013, pp. 114-119

[5] A. Passito, E. Mota, R. Bennesby, P. Fonseca, Institute of Computing Federal University of Amazonas Manaus, AM - Brazil: "AgNOS: A Framework for Autonomous Control of Software-Defined Networks", 2014 IEEE 28th International Conference on Advanced Information Networking and Applications, pp. 405-412

[6] J. Crowcroft, M. Fidler, K. Nahrstedt, R. Steinmetz: "Is SDN the De-constraining Constraint of the Future Internet?", ACM SIGCOMM Computer Communication Review, Volume 43, Number 5, October 2013, pp. 13-18

[7] R. Jain, S. Paul, Washington University: "Network Virtualization and Software Defined Networking for Cloud Computing: A Survey", IEEE Communications Magazine, Novembre 2013, pp. 24-31

[8] B. Lantz, B. Heller, N. McKeown: "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks", Hotnets '10, October 20-21, 2010, Monterey, CA, USA, pp. 1-6

[9] Mininet - mininet.org

[10] D. D. Clark, C. Partridge, J. C. Ramming, J. T. Wroclawski: "A Knowledge Plane for the Internet", in SIGCOMM '03. New York, NY, USA: ACM, 2003, pp. 3-10.

[11] R. Boutaba, Y. Iraqi, A. Mehaoua: "A Multi-Agent Architecture for QoS Management in Multimedia Networks", Journal of Network and Systems Management, vol. 11, pp. 83-107, 2003, 10.1023/A:1022497125456

[12] M. Wooldridge: "An Introduction to Multiagent Systems", 2nd ed. Wiley, 2009

[13] FIPA: "Fipa ACL Message Structure Specification", Foundation for Intelligent Physical Agents, http://www.fipa.org/specs/fipa00061/SC00061G.html, 2004

[14] N. J. Vieira: "Máquinas de Inferência para Sistemas Baseados em Conhecimento", Ph.D. dissertation, Pontifícia Universidade Católica do Rio de Janeiro, 1987

[15] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, S. Shenker: "NOX: Towards an Operating System for Networks", SIGCOMM Comput. Commun. Rev., vol. 38, pp. 105-110, July 2008

[16] X. Liu, X. Yang: "NetFence: Preventing Internet Denial of Service from Inside Out", in SIGCOMM '10. New York, NY, USA: ACM, 2010

[17] F. Soldo, K. Argyraki, A. Markopoulou: "Optimal Source-based Filtering of Unwanted Traffic", in IEEE/ACM Transactions on Networking, 2012

[18] R. B. Braga, E. M. Mota, A. P. Passito: "Lightweight DDoS Flooding Attack Detection using NOX/OpenFlow", Local Computer Networks, Annual IEEE Conference on, vol. 0, pp. 408-415, 2010

[19] X. Liu, X. Yang, Y. Lu: "To Filter or to Authorize: Network-Layer DoS Defense Against Multimillion-Node Notnets", in SIGCOMM '08. New York, NY, USA: ACM, 2008, pp. 195-206

[20] K. Kirkpatrick: "Software-Defined Networking", comunications of the ACM, settembre 2013, vol. 56, no.9, pp.16-19

[21] M. Sridharan et al.: "NVGRE: Network Virtualization Using Generic Routing Encapsulation", IETF Draft draft-sridharan-virtualization-nvgre-03.txt, Aug. 2013, http://tools.ietf.org/html/draft-sridharan-virtualization-nvgre-03, pp. 17

[22] S. Paul, R. Jain: "OpenADN: Mobile Apps on Global Clouds Using OpenFlow and Software Defined Networking!", 1st Int'l. Wksp. Management and Security Technologies for Cloud Computing, Dec. 7, 2012

[23] H. Kim et al.: "The Evolution of Network Configuration: A Tale of Two Campuses", Proc. 2011 ACM SIGCOMM Conf. Internet Measurement Conf., New York, NY, 2011, pp. 499-514.

[24] S. Sundaresan et al.: "Broadband Internet Performance: A View from the Gateway", Proc. ACM SIGCOMM, Toronto, Ontario, Canada, Aug. 2011

[25] A. Voellmy, H. Kim, N. Feamster: "Procera: A Language for High-Level Reactive Network Control", Proc. 1st Wksp. Hot Topics in Software Defined Networks, New York, NY, 2012, pp. 43-48

[26] M. Chetty et al.: "You're Capped: Understanding the Effects of Bandwidth Caps on Broadband Use in the Home", Proc. 2012 ACM Annual

Conf. Human Factors in Computing Systems, CHI '12, New York, NY, 2012, pp. 3021-30